@(#)top:

Issue #14
Version 1.0
November 1996

@(#)contents: Table of Contents

Features
   6. The Commodore Telnet BBS by Bo Zimmerman
      (Reference: net)
        In this age of internetworked computer systems, is the
        Commodore left out? No way, as Bo Zimmerman describes how to
        coax your Commodore BBS system to play the networking game.
        Bo shows how to set up your BBS so that Internet users can
        "telnet" to your BBS from anywhere on the 'Net.
   8. Menu Toolbox III by Jeff Jones
      (Reference: toolbox)
        You've got this neat idea for a game, utility, or productivity
        application. The engine is complete and working, but the user
        interface is a mess. Do you scrap the project because you're
        not up to the task of writing a whole UI engine? Nonsense. Jeff
        presents a rich set of functions and subroutines to tame that
        killer application.
  11. The CMD Nirvana: The Guts and Glory by Todd Elliott
      (Reference: hw)
        Has your computer system started looking like the multi-headed
        beast from a "B" movie? Are you tired of having so many items
        on your desk? Do you envy IBM PC owners with their all-in-one
        computer? Well, if you answered YES! to any of the above, let
        Todd show you his souped up C128DCR. Learn how you, too, can
        "upgrade" your computer system and refine its image.
  13. Jim Butterfield: The Commodore Guru - An Interview by Jim Lawless
      (Reference: jb)
        Jim Butterfield has long been associated with the Commodore
        computer system, from the days of the KIM-1 to the present.
        Many of us learned machine language through Jim's articles or
        books, while most have benefitted from his early work on
        creating memory maps and documenting KERNAL routines for the
        Commodore line. Jim Lawless talks to the ubiquitous Commodore
        Guru.

Columns
   4. Hi Tech Trickery by Alan Jones
      (Reference: trick)
        In part II of Alan's "Heavy Math" series, he moves right into
        Linear Programming and how to accomplish it on the C64. If you're
        still not sure what LP math is, read on, as you'll be surprised
        at which everyday problems fall into this category of mathematics.
  15. Hacking BASICs by Richard T. Cunningham
      (Reference: basic)
        Even as more and more programmers take up the ML flag and wave it
        proudly, there are many who either use BASIC entirely, or prototype
        pieces of code in BASIC before converting to ML. Richard outlines
        some common "gotchas" in the ever-present programming language.
  17. Twiddling the Bits by Ward Shrake
      (Reference: bits)
        OK, VIC-20 enthusiasts, listen up. Resident VIC-20 cartridge expert
        Ward Shrake details exactly how the VIC-20 and its cartridges work
        together to allow the user to play games and use applications on
        cartridge. Ward details how to archive your collection of VIC
        carts, as well as how the computer recognizes and executes code on
        a cartridge.

Departments
   1. The (cough, cough) Hacking Editor
      (Reference: editor)

   2. Input/Output
      (Reference: io)
   3. Newsfront
      (Reference: news)
   5. Hacking the Mags
      (Reference: mags)
   7. UseNuggets
      (Reference: usenet)
   9. FIDO's Nuggets
      (Reference: fido)
  10. The Hacking Review
      (Reference: review)
  12. Hack Surfing
      (Reference: surf)
  14. Commodore Trivia
      (Reference: trivia)
  16. ? DS, DS$: rem The Error Channel
      (Reference: error)
  18. The Next Hack
      (Reference: next)
  19. Hacking the Code
      (Reference: code)

@(#)legal: Commodore Hacking Legal Notice

Commodore and the respective Commodore product names are trademarks or registered trademarks of ESCOM GmbH or Visual Information Services Corporation. Commodore Hacking is in no way affiliated with ESCOM GmbH or Visual Information Services Corporation (VISCorp), owners of said trademarks. Commodore Hacking is published 4 times yearly by:

Brain Innovations Inc.
10710 Bruhn Avenue
Bennington, NE 68007

The magazine is published on on-line networks free of charge, and a nominal fee is charged for alternate mediums of transmission.

Permission is granted to re-distribute this "net-magazine" or "e-zine" in its entirety for non-profit use. A charge of no more than US$5.00 may be charged by redistribution parties to cover printed duplication and no more than US$10.00 for other types of duplication to cover duplication and media costs for this publication. If this publications is included in a for-profit compilation, this publication must be alternately available separately or as part of a non-profit compilation.

This publication, in regards to its specific ordering and compilations of various elements, is copyright (c) 1995-96 by Brain Innovations, Incorporated, unless otherwise noted. Each work in this publication retains any and all copyrights pertaining to the individual work's contents. For redistribution rights to individual works, please contact the author of said work or Brain Innovations, Inc.

Brain Innovations, Inc. assumes no responsibility for errors or omissions in editorial, article, or program listing content.

@(#)info: Commodore Hacking Information

Commodore Hacking is published via the Internet 4 times yearly, and is presented in both ISO-8859-1 and HTML versions. This and previous issues can be found at the Commodore Hacking Home Page (http://www.msen.com/~brain/chacking/), as well as via FTP (ftp://ccnga.uwaterloo.ca/pub/cbm/hacking.mag/)

In addition, the Commodore Hacking mail server can be used to retrieve each issue. To request a copy of an issue, please send the following electronic mail message:

To: brain@mail.msen.com
Subject: MAILSERV
Body of Message:

```
help
catalog
send c=hacking13.txt
quit
```

To retrieve a PKZIP 1.01 archive of the individual articles in Commodore Hacking, request the file c=hacking13.zip

To subscribe to the Commodore Hacking and receive new issues as they are published, add the following command to you MAILSERV message prior to the quit command:

subscribe c=hacking Firstname Lastname msglen

(msglen is largest size of email message in line you can receive. Each line is roughly 50 characters, so 600 lines is about 30000 bytes. When in doubt, choose 600)

example:

subscribe c=hacking Jim Brain 600

Although no fee is charged for this magazine, donations are gladly accepted from corporate and individual concerns. All moneys will be used to defray any administrative costs, subscribe to publications for review, and compensate the individual authors contributing to this issue.

As part of a magazine promotion, Commodore Hacking Issue #12 was professionally laid out on printed format. These printed copies are for sale.

If you can not obtain Commodore Hacking through any other means and wish to purchase a copy on disk or would like to purchase the professionally printed Issue #12, please address a check or money order to "Jim Brain" and mail to:

Jim Brain
10710 Bruhn Avenue
Bennington, NE 68007

Disk copies of each issue: USD$5.00
Professionally printed copy of Issue #12: USD$6.00

All prices cover only duplication and materials and include shipping in the United States. For disk copies, please specify format:

Computer	Disk Size	Capacity	Notes
CBM/PETSCII	5.25 inch	170 kB	1541 format
		340 kB	1571 format
	3.50 inch	800 kB	1581/FD2000 format
		1.6 MB	FD2000/FD4000 format
IBM/ASCII	3.50 inch	720 kB	Double Density
		1.4 MB	High Density

Any persons wishing to author articles for inclusion in Commodore Hacking are encouraged to view the submission guidelines on the WWW (http://www.msen.com/~brain/pub/c-hacking-submit.txt) or via the MAILSERV server (send c-hacking-submit.txt).

==========

@(#)rch: Reading C=Hacking

Starting with Issue 11 of Commodore Hacking, the new QuickFind indexing system is utilized to aid readers of the text version in navigating the magazine. At the top of each article or other important place in the magazine, a word prefixed with a special string is present. (See the title of this article for an example.) Throughout the magazine, if an article is mentioned, it will be followed by a reference string. For example, if we mentioned this article, we would add (Reference: rch) after the name. By using your favorite editor's search function and searching for the string after the word "Reference:", prefixed by the magic prefix string, will move you directly to the article of choice. To merely skip to the next article in the magazine, search only for the magic prefix string.

Some handy indexing strings possibly not referenced anywhere are:

top top of issue
bottom bottom of issue
contents table of contents
legal legal notice

For those with access to a UNIX system, the command "what" can be run on the issue, which will result in all the article titles being printed.

A slightly different magic prefix string "@(A)" is used to delimit sub-topics or main heading in articles. The text after the magic string differs depending on article content. For the Input/Output column (Reference: io), the text after the magic prefix will either be "c" for comment, or "r" for response. In features and columns, a number after the prefix indicates the ordinal of that heading or sub-topic in the article. If a specific sub-topic is referenced elsewhere in the article, a sub-topic reference will be indicated. A reference to "@(A)r" would be written as "(SubRef: r)".

As time goes on, the role of this indexing system will be expanded and changed to ease navigation of the text version, but minimize the clutter added by these extra items.

==========

@(#)editor: The Hacking Editor
by Jim Brain (j.brain@ieee.org)

Sometimes, it's important to look back and see how far we've come. The following story comes to mind:

A young boy sits in the living room and flips earnestly through a Montgomery Wards catalog looking for some item. The year is 1983. At last he finds the item and presents the book to his father, who is reading a periodical in his easy chair. "Dad," the boy begins, "I want to buy one of these with my savings." The father, startled upon hearing of such a prospective purchase, looks up and reaches for the catalog. "What is it you want to buy?" he asks. "That video game on the top of the page is what I want," the boy explains. The father looks at the pertinent page and notices a glossy picture of an Atari VCS2600 console system, complete with options. Frowning, the father raises his head and look in the boy's eyes. "Son," he starts, "I am not going to let you buy one of these video game systems. All they are good for is playing games, and that's too much money to spend to buy a game." The boy protests, stating that "all his friends" own one and that it the "thing" to own today. The father, known for being stubborn, refuses to budge on the issues, but concludes the exchange by handing the catalog back and saying, "If you want to buy a machine that plays games, buy one of those new computer systems. That way, you can play games with it and also use it for other things when the games get old and boring." The boy takes back the book and sulks for a while as he flips through the pages. As the hurt wears off, he notices a section near the video console page that shows off those new computer systems his Dad referred to. At first, the kid's eye is drawn to the shiny silver Texas Instruments TI-99/4 computer system pictured in the catalog. He is about to jump up and again hand the catalog to his Dad when he realizes the "new-fangled" item is priced at $322.00. His heart sinks, for his savings account only holds a bit over $250.00 and the machine looked so impressive. So, beaten again, the young boy flips the page and resigns himself to never owning anything "cool". However, the next page pictures a different computer system and a quick check confirms the price is within budget: $233.00. The computer isn't as impressive looking as the TI, but the boy will not be without a "video game", and this fits the bill.

Needless to say, the computer was a Commodore VIC-20, and the boy bought a few games for the unit, including a Space Invaders clone and a Pac-Man clone. As the father predicted, the boy lost interest in the unit after a while and packed the system away. However, as the boy entered 7th grade, he again pulled the unit out when he learned that one of his classrooms was equipped with Commodore VIC machines. His interest in computers as tools started there and grew with the years.

As I finish my first year of editorship of Commodore Hacking, I am looking back at the events that have occurred in the last year and those that have occurred over the years since I first learned about Commodore computers. Commodore owners have come from 3.5kB and 22 by 23 screens with the VIC-20 to CBM machines with features like multiple megabytes of RAM, 33.6 kbps FaxModems, gigabyte hard drives, 8-20 MHz operation, and a host of other options. No, I don't think Commodore computers can solve all the world's problems. However, they and their owners should be commended on their loyalty and dedication to the market and to the advances that have kept the machines out of closets and dumpsters. While I won't doubt that there are more IBM PC clones in the world today, I wonder how many PC units are resting under tons of refuse in the city dump.

Here at Hacking Headquarters, I am impressded by what we have accomplished with the publication, but I have already outlined improvements that can be made and things I didn't quite get implemented this past year. As always, your letters and comments are always appreciated. The publication depends

on reader feedback to ensure that covers subjects of interest to the Commodore enthusiast. Of course, some things, like the technical focus of Commodore Hacking, define the magazine and its place among the various Commodore publications. However, even that can be continually improved. So, as you look back on the past year of Commodore usage, take a look at our progress or lack thereof and send us a note, if only to tell us to change nothing. Remember, we can't increase the publication's usefulness to you if we don't know where it currently falls short.

As for the boy in the above story, I think he's come a long way since that fateful day in 1983. He no longer thinks TI's look better than CBM's. In fact, I think he has earned an impressive reputation as a Commodore advocate. Then again, I might be a bit biased, so you be the judge. The boy in the story was a youngster named Jimmy. Jimmy Brain.

Enjoy YOUR magazine,

Jim Brain (j.brain@ieee.org)
editor

==========

@(#)io: Input/Output

Obviously, Commodore Hacking depends on the comments and article submissions from the Commodore community to flourish. Everyone sees the articles, but let's not forget those comments. They are very helpful, and every attempt is made to address concerns in them. Address any comments, concerns, or suggestions to:

Commodore Hacking
10710 Bruhn Avenue
Bennington, NE 68007
j.brain@ieee.org (Internet)

@(A)c: Hey! You Characters! Sit Down!

From: Adam Vardy (abe0084@InfoNET.st-johns.nf.ca)

Dear C=Hacking,

In the last issue there was some source code for printing very big numbers. This source code is all in uppercase. This seems to be true whenever source code is included in the magazine. I am wondering why that is. This makes it rather difficult for me to extract the source and put it into a form that my assembler can deal with. I can't load the source right into Power Assembler. It only accepts lowercase code. It is puzzling to me why you do this, because I would think any assembler that accepts plain text would work this way too.

Another thing is this. In the last issue one of the uuencoded files in the magazine was dim4. The source code for the included files is for the Merlin Assembler. OK. So I try to read these files. I'm having problems with this. If I try to More them in ACE, I can't. It's unreadable. They seem to be text, but however I try to read them, I get weird characters or other stuff. Loading them into a word processor or into ZED, or anything doesn't work.

I don't have Merlin. But I would think it must have some way to save plain text source. That way, everyone can at least read it, right?

@(A)r:
Code is printed in the magazine as it is received by Commodore Hacking. The only formatting done to source code in articles and columns is to indent each line 3 spaces. The source code to which you refer above was in a USENET posting and was captured from the comp.sys.cbm newsgroup in uppercase. Our theory is that some folks who upload code to the Internet do not do an PETSCII-ASCII translation, which would cause the effect of switching all lowercase characters to uppercase. However, we are not certain that there all assemblers expect lowercase, which is why we do not try to alter case of source code.

As for your second problem, we accept part of the blame. We are attempting to obtain allof the source code used in the publciation in ASCII or PETSCII format. However, a number of assemblers, including Turbo-Assembler, do have an internal format that is neither ASCII nor PETSCII. Merlin may also have such a format. However, we are unfamiliar with Merlin, so it may not have an option to output code in ASCII or PETSCII, as Turbo-Assembler does. Our suggestion is to contact the author of the article directly and ask for an ASCII copy of the source and accept our apologies.

@(A)c: A Plea for Information

From: MICHAEL I DEMING <m.deming@genie.geis.com>

Dear C=Hacking,

An article or series of articles on the 80 column chip would be very helpful e.g. how to use sprites, a screen dump and other things like that. If I knew how to do this I would write the articles but I don't so I am begging for any info on this chip.

@(A)r:
Always check back issues of Commodore Hacking for prior articles on topics. See Commodore Hacking Information (Reference: info) for directions on how to access back issues. VDC information is included in Issue 1 as part of Craig Bruce's "Simple Hires Line Drawing Package for the C128" and as part of Craig Taylor's "An In-Depth Look at the 8563 Video Chip on the C= 128" in issue 2. As for the other topics, other articles in Commodore Hacking touch on those issues, but we always appreciate new articles dealing with these topics.

@(A)c: All I Can Say is Wow!

From: George Szaszvari <gsz@dial.pipex.com>

In Commodore Hacking #13 Preface:

>Whew! Folks, here is the long awaited Issue #13 of Commodore Hacking.
>Hacking Headquarters has produced an issue overflowing with technical
>articles sure to satisfy even the most discerning Commodore enthusiast.
>In fact, this issue is OVERFLOWING with 384 kB of material, so empty out
>that mailbox. Here it comes...

Yeah, a real BUMPER issue, thanks!

@(A)r:
Well, the size is a both a blessing and a curse. While we are happy about the number and diversity of articles, we know there are those who can't handle a large issue like #13, so we are trimming the size a bit from #14. However, thanks for the comments.

@(A)c: Speaking of Kudos!

From: Brett Tabke

Dear C=Hacking,

Thank you! One of, if not THE, best issues yet!

I can't thank you enough for all the work you've done here Jim. Between Hacking, The FAQ, and the CBM product documetation, you have put out more valuable information in 6 months than most of the pay magazines to in their lifetime. The CBM products listing is a rare treasure that every CBM owner should take time to read.

@(A)r:
We don't know what to say. We're just happy that everyone stood by us during the move and the delay in getting #13 out. By th way, for those who have not seen. The CBM Products List to which Brett Tabke refers is available as "cbmmodel.txt" on the MAILSERV and through the WWW. (http://www.msen.com/~brain/pub/cbmmodel.txt) If you prefer to wait, an updated copy will be presented in Commodore Hacking #15.

@(A)c: Who's Got the Right Copyright?

From: Ruth Hackley (fgm@rosenet.net)

Dear C=Hacking,

I am Ruth Hackley, Ron's wife, and newsletter editor for the L.C.C.U.G. in Eugene. Are there any portions of C=Hacking that can be used in the newsletter. We plan to provide the magazine on disk to our library as well.

@(A)r:
The entire publication can be redistributable as a complete work, as explained in the Commodore Hacking Legal Notice (Reference: legal). As well, individual articles can be reprinted with permission from the author of the article. Commodore Trivia is a special case. Permission has been given by the author to reprint Commodore Trivia in newsletters and

publications, just as _Commodore World_ and this publication do.

@(A)c: A Little Lacking

From: Scott Brockway <fungus@eskimo.com>

Dear C=Hacking,

I thought the magazine was sort of a transactor replacement. When I downloaded issue 12 of C=Hacking I was not just a bit disappointed. I like all the technical stuff and now I fear the mag is no longer for me.

Hey C=H, Put the code back in!

@(A)r:
We are sorry you felt this way. It was not, nor is it currently, our intention to leave technical readers without articles of interest. We ask that you take a good look at the magazine. Others have initially thought the magazine lacked technical content, but later determined that the style of some articles had changed and the technical articles were separated by a few less technical offerings. However, be aware that due to the time and space constraints we are trying to fit Commodore hacking in will reduce the number of technical articles by 1 or possibly 2 each issue over the old issues. Also, some technical articles do not contain any code pieces.

In the case of Issue 12, we were forced to redo the issue after an important technical article from a semi-regular writer did not appear. The resulting rework might have shown through. We hope Issue 13 provided enough technical content. Our most technical readers might find the current issue a bit light on content, due to problems associated with Commodore Hacking's recent relocation efforts. We ask that the technical readers bear with us as we ramp up in our new location.

One of our continual problems, and one of Craig Taylor's (the previous editor) as well was finding quality technical articles to publish. One way to solve the problem is to delay the publication date, a tactic Craig might have used. However, we are trying to get back on a stable schedule. So, if you are up to it, write up a technical piece for inclusion in the next issue.

@(A)c: A Bit of Their Own Medicine

From: wanderer_rtc@usa.pipeline.com (R. T. Cunningham)

Dear C=Hacking,

I just read CHacking #13 and was very impressed with the amount of work put into it. While I was reading the HTML tutorial, I came to a not quite brilliant idea.

While working on an HTML viewer, take time to see if you can work standard C/G viewing. I would love to do up a bunch of web pages using C/G with the familiar disclaimer up on top, previously posted by someone else and plagiarized by me, <best viewed using a Commodore 8 Bit Computer>. This would please quite a few people (besides me of course). Who says we have to be able to see 256 colors!

@(A)r:
Oh, do we detect a bit of revenge here? I am sure Jim Brain can arrange C/G graphics in the HTML viewer he is working on, but we can see a problem:

The eventual goal of the HTML viewer is to grow the application into a full fledge WWW Browser. If a site uses C/G graphics exclusively, the large number of Commodore enthusiasts that view WWW sites with non-CBM graphical browsers will not see the C/G graphics.

In addition, some would-be Commodore enthusiasts who explore these C/G graphics enabled sites might leave thinking we are snobs and never return. We don't need that.

We think a slightly altered suggestion is better. When designing the "<IMG>" tag in he HTML viewer, allow it to understand the optional attribute "CBMSRC". That way, a Commodore site can safely display graphics to non-CBM browsers and still put the "Best Viewed with a C64" on the page. All the graphics on the page would then be specified by an image tag looking like: <IMG SRC="junk.gif" CBMSRC="better.cg">. The CBMSRC attribute could then be used on a CBM browser to display the alternate C/G graphic. A non=CBM browser would ignore the CBMSRC attribute, and a tag like: <IMG CBMSRC=...> would be ignored by the browser as well, since it contains no SRC attribute. And, best of all, if you really wanted to keep the Netscape browsers out, simply put <IMG SRC="better.cg">.

@(A)c: UU what?

From: Cameron Kaiser <ckaiser@sdcc13.ucsd.edu>

Dear C=Hacking,

I noticed that the last C-Hacking had a number of uuencoded
files concatenated to each other. This presents a problem in unix
because a number of crufty uudecoders don't recognize them together
(only the first one). I have made a PERL script that folks can use to
fix this. I'll leave it in
ftp.armory.com/pub/user/spectre/UNIX/multiuu.pl

@(A)r:
We appreciate the utility. If someone wants to take advantage of this
helpful utility, simply FTP the file to your local UNIX account and execute:
chmod u+x multiuu.pl
Then you should be able to uudecode multiple part files with this program.

@(A)c: Comfortable Reading

From: rbthomas@freenet.edmonton.ab.ca

I have d-loaded a few issues of C= Hacking and converted them to GEOS and
looked at them with geoWrite but that is a clumsy process. Also, I
understand some of the issues have program files in them and these need
to be extracted for use. How does a person go about this?
Thanks in advance for any help you can offer.
I have a 1581 and an FD-2000 drive so the space isn't a problem with
storing the file. I would like to read it in comfort also. TWS won't
handle it.

@(A)r:
To help people who can't handle the large size of Commodore Hacking,
each issue is now available as an archive of individual articles and already
decoded executable files. Commodore Hacking Information (Reference: info)
has information on how to obtain and dowload the archive files.

==========

@(#)news: Newsfront

@(A): Commodore Hacking CANCELED!

As many know, one of the distribution mediums for Commodore Hacking is
USENET, which is a services that operates primarily on the Internet.
After dutifully "posting" Issue #13, Hacking Headquarters was informed
that the posting had been "canceled". Since normally only the original
poster can delete or "cancel" a posting, we were alarmed. The culprit
turned out to be a automated program that had been installed on USENET
since Issue #12 was published. The program, called a "cancel robot" or
"cancelbot" for short, had been created by an individual and installed
on one USENET node. This specific cancelbot watches for large postings
to non-binary newsgroups (newsgroups that do not have "binary" or
"binaries" in their names) that contain large UUencoded binary files.
It then "forges" a cancel message by masquerading as the original poster.
Since USENET contains very little security, this notion of "forging"
postings can be done quite easily.

Without detailing the technical side of USENET, suffice it to say that
a posting from one site immediately begins its journey to all other sites
on USENET, being replicated along the way. The cancelbot canceled the
posting immediately after it showed up on its server. The cancel message
then began its journey to each server, canceling the article at each site.
Given the propogation delay of USENET, the posting was up long enough for
some readers to acquire it, but not everyone. Therefore, a pointer to
the location of the issue was posted later.

Along with the pointer posting, Commodore Hacking asked the USENET newgroup
comp.sys.cbm how it should ahndle the situation. We offered three
suggestions and asked for comment. They were:

1. Request an exclusion for the publication from the cancelbot
2. Publish only a pointer to the location of the new issue.
3. Publish the issue stripped of UUencoded binary files.

Suggestions 1 and 2 received an equal number of votes, with suggestion 3
receiving a couple of votes and 1 person voting to split the issue into
multiple parts. Needless to say, the issue is still undecided.

Therefore, for the current issue, we have requested an exclusion from the USENET cancelbot. However, since most readers can now access the publication via the World Wide Web, Electronic Mail, and FTP, we are considering publishing only an announcement in the future. The announcement will highlight the newest issue and detail where to obtain it.

Some of the survey respondents mentioned that a few readers may only have access to USENET as a means of receiving the magazine. If you are one of those folks, PLEASE WRITE US! We are continuing to post the entire issue on USENET for your benefit and may not continue to do so if we don't hear from you. The postal mail address is located in this issue (Reference: legal). If Commodore Hacking does not receive any aformentioned letters or objections, we will consider posting only new issue announcements in the future. Doing so would relieve some burden on the USENET system, which was really not designed to handle a posting of C=Hacking's size. If you have any questions about the potential impact of this change, please mail us at Hacking Headquarters.

@(A): The The Underground Went South!

Shortly after the release of Commodore Hacking #13, Underground Magazine editor Scott Eggleston informed his subscribers that changes in his life had forced him to cut back on the time devoted to publishing, and that he was merging The Underground with the newly announced LOADSTAR LETTER commercial publication. As reported in the last issue, Scott will co-edit the new expanded LOADSTAR LETTER with Jeff Jones and unfilled Underground subscriptions will be filled with LOADSTAR LETTER subscriptions on a 1 for 1 basis.

@(A): Get Yer QWKRR128 5.0! Read All About It!

In Commodore Hacking #13, Gaelyne Gasson (formerly Gaelyne Moranec), described how to read Internet electronic mail offline by using an offline mail reader like QWKRR128. Hot on the heels of that article,Rod Gasson has announced the availability of a beta of version 5.0 of the QWKRR128 software. Available to all registered users of QWKRR128, the beta includes a number of enhancements and bug fixes, including:

1. Supports full 255byte character sets.
2. Reads messages of ANY length (including the ability to print, export, or small.dat them).
3. Separate VIP & TWIT lists.
4. UUdecodes messages of any length as long as the UUencode is in a single message. (It won't decode if the file is split over several messages).
5. Decodes MIME messages (Base64). Same restriction as UUdecodes.
6. Added keyboard tables so it can be configured to 'international standards'.
7. Updated the 'auto-netmail' routines to include Internet Email as well as fido netmail.
8. Improved the address book so it will handle both the fidonet format addresses and email style addresses.
9. Added the ability to ATTACH files to a message or reply. These attaches are limited in length to about 8megs (the max that the QWK format can handle).
10. Improved the routines to detect a valid Q.NDX file (the name of the new QWKRR index file). This means these no longer have to be manually scratched prior to reading a new mail packet.
11. Added code so that you no longer have to quit QWKRR in order to read a different mail packet.
12. Improved the macros so that whole words can be used as a 'trigger'. This can be used as a 'simple' spell corrector. (eg, a macro such as "Ismeal=Ismael" will ensure that you'll never transpose the "a" and "e" in Ismael again.
13. Added a 'scrap macro' that can be defined and used while in the editor itself, without the need to add it to your macro file).
14. Added code so that quote initials can be changed 'on the fly' (This is useful when quoting from Email messages where the default initials are often meaningless).
15. Added time/date stamping to the zipped REP packets. (Some BBS's didn't like having the same date/time stamp on all mail packets).
16. Improved access speed by about 3 times for Ramlink users.
17. Improved tagline handling. You can have up to 10,000 taglines in one category. Tagline files are now numbered from .000 to 999.
18. Several cosmetic changes and bug fixes.

The new beta version is available from the 221b Baker Street BBS in the US and GEOZ BBS in Australia and at:

ftp://ccnga.uwaterloo.ca/pub/cbm/INCOMING/telecomm/qwkrrv5b.lzh

ftp://hal9000.net.au/pub/cbm/qwkrr/qwkrrv5b.lzh

More complete information is available at:

http://www.hal9000.net.au/~moranec/qwkrr10.html

@(A): Centsible Software Address Update (And Some Bad News)

The folks at Centsible Software have moved! Well, they are still at the same location, but they have "electronically moved", to sprynet.com. The new information appears below:

Centsible Software
PO Box 930
St. Joseph, MI 49085
616-428-9096 (Orders and Information 12-6pm EST)
616-429-7211 (Bulleting Board System and Facsimile)
Cents@sprynet.com (Internet Contact)
http://home.sprynet.com/sprynet/cents (WWW URL)

On a sad note, Centsible contacted Hacking Headquarters later to note that they will soon be closing down Centsible Software. All outstanding orders will be completed, and the closure won't happen immediately, but soon Centsible will be all but a memory.

@(A): Brace Yourself, There's More

Software Support International (SSI), has recently decided that they will start focusing more on IBM compatible sales. In an initial announcement, SSI indicated that they would no longer carry Commodore products after January 1, 1997, but later announced that they would continue to sell products as long as existing stock holds up. However, they will no longer promote the CBM or Amiga product lines. The latest catalog from SSI will be the last to carry CBM and Amiga merchandise.

@(A): And Now for Some Good News

Arkanix Labs, a West Coast hardware and software developer, recently announced that they had acquired Threshold Productions International. Petar Strinic (note the 'a' in Petar) announced that the acquisition would expand their presence in the market. Arkanix Labs develops for the MS-DOS and C64/128 systems, and is working on SuperCPU products. To find out more about the company, visit their WWW Site at: www.arkanixlabs.com or contact Mr. Strinic at: petars@arkanixlabs.com

@(A): The Underground To Live on in LOADSTAR LETTER

Scott Eggleston, editor of The Underground, has recently sent notice to all Underground readers that recent changes in his life have prompted him to discontinue publishing of The Underground. Determined that Underground subscribers would not be left in the lurch, Scott has arranged for each subscriber to receive issues of the newly launched commercial LOADSTAR LETTER publication (See Newsfront in C=H #13). While the size of each issue will be smaller, the issues will arrive monthly, as opposed to The Underground's bimonthly schedule.

Scott, not bowing out of publishing entirely, will be brought on as Associate Editor of LOADSTAR LETTER. Subscribers should expect to receive their first issue of the merged publication at the same time The Underground #15 would have arrived.

As well, Scott will continue to offer back issues of The Underground for US$2.50 per issue. The Underware disk will no longer be available from Eggleston, although reader Tom Adams (tom.adams@neteast.com) has agreed to copy the disks as long as he is able to do so.

If you need to contact Scott Eggleston, you may do so at:

egglest1@cougarnet.byu.edu

@(A): Video Interface Computer (VIC) Software Available

Ghislain deBlois (dh374@freenet.carleton.ca) is currently releasing a number of games and utilities for the avid Commodore VIC-20 user community. Intending to release them in "cassette/disk of the month" fashion, deBlois' first installment contains:

o Meteor Zone
o Mini Assembler
o Vicfall!
o Ringside

Future installments promise titles like: Realms of Quest, Ice Hockey, Bosing, and Screen Magic (a multi-color hi-res drawing program). All programs are written for the unexpanded VIC-20 computer system (VC-20 in Europe) to better serve VIC enthusiasts. For more information, contact deBlois at his electronic mail address.

@(A): Get on the Super CPU list!

Brett Tabke, of PHD Software, has created a mailing list for owners of the CMD SuperCPU accelerator cartridge. To subscribe to the list, send a message to:

listserve@giga.or.at

with a message body of:

subscribe super-cpu firstname lastname

@(A): Mailing Lists, Take Two!

For those who live in on the OTHER side of that little pond from the US, or if you just want to keep up on the developments there, there is a new electronic mailing list for European 64 enthusiasts. Simply send email to:

listserv@lentil.demon.co.uk

with a subject of:

MAILSERV

and a body of:

subscribe 64EUROPE
END

The list address is: 64europe@lentil.demon.co.uk

@(A): It's Better than Novaterm 9.6!

Nick Rossi has recently announced that patch "A" for the latest version of Novaterm (9.6) is now available. Citing the help of early purchasers of the product, Nick noted a list of bug fixes that have been incorporated in the new release, including:

* Estimated file length omitted from Zmodem upload, to avoid confusing BBS's.
* The "funny ASCII" problem when capturing to the buffer in ANSI or VT102 mode has been fixed.
* Due to the above, you should remove the ".opt ansi" line from all your scripts and recompile them.
* A bug in the script function that checks for incoming strings was fixed.
* The "Save buffer when full" option now works properly with hardware flow control.
* A bug in the recovery function of the BBGRam driver was fixed.
* A bug in one ANSI screen clearing function was fixed.
* Several errors in the font81.ANSI character set were fixed: ASCII 160 was changed to á, and the fractions &188;, &189;, and &190; were put in their proper order.
* Typing a shift-space now sends a space, rather than the á character.
* Color was added to the VT102 terminal emulation.
* A real-time clock driver was added to read from the CMD SmartMouse and SmartTrak.
* Functions in the text editor were fixed: loading/saving in the buffer and changing the device number.

If you have already purchased Novaterm 9.6, you can receive a free upgrade by making a backup of your master disk and sending the master disk back to:

Nick Rossi
10002 Aurora Ave. N. #3353
Seattle, WA 98133 U.S.A.

Copies of the latest release are available in either 1541 or 1581 disk formats for USD$29.95 plus USD$1.50 for shipping and handling. The software is accompanied by a 90 page user manual.

For more information on the upgrade or Novaterm in general, visit the Novaterm 9.6 WWW Site: http://www.eskimo.com/~voyager/novaterm.html or

contact Mr. Rossi via email at: voyager@eskimo.com.

@(A): Who's Got the Rights to the Commodore 8-bit line?

That is a very good question. Ever since Commodore was sold to ESCOM GmbH, Commodore 8-bit users have wondered who owns the intellectual rights to the Commodore 8-bit line of computers. Now that ESCOM has declared bankruptcy and initiated the sale of the Amiga line to US based Visual Information Service Corporation (VISCorp), CBM enthusiasts are even more curious. The sale, evidently drawn up before the bankruptcy declaration, would place Amiga technology into the hands of VISCorp, which was started by several ex-Commodore engineers. However, even if or when the deal is finalized, who owns the rights to the CBM 8-bit line may still be a mystery.

@(A): New Address for Meeting 64/128 Users Through the Mail

Tom Adams, president of the mail correspondence club called Meeting 64/128 Users Through the Mail, asks that all electronic correspondence for either him or the club be addressed to:

   tom.adams@neteast.com

@(A): Complete you Transactor Collection!

Karl Hildon, one of the producers of the now-defunct _Transactor_ publication, has announced that he is now able to provide electronic copies of any issue of the technical magazine for USD$5.00. Issues can be scanned in the format of your choice, and will be electronically mailed to the purchaser.

To order, mail Karl Hildon your Visa Card account number (visa only) and expiration date and issue number request to karl@inforamp.net. If you need to consult an index first, there is one located at: http://vanbc.wimsey.com/~danf/cbm/transactor.idx

Mr. Hildon also mentioned that if demand warrants, he will also make available the _Transactor_ companion disks.

Also, Mr. Hildon has announced that he can also make available copies of _The Inner Space Anthology_ for USD$20.00 or CAN$20.00. Follow the above directions to obtain copies of this out of print resource.

@(A): "Ultimate" Demonstration Contest Announced

Commodore Zone and Tim Wright have announced the Commodore 64 Golden Fleece Award 1997 contest. An award of $100.00 will be presented to the author of a demo program that represents the best in demo construction and captures the spirit that have made demos a staple of Commodore history. The deadline for entry is February 1, 1997, and entries should be sent to "Binary Zone P.D. Entries must fit on a single side of a 1541 disk, and will be judged as complete works. Authors can enter as many works as they wish. Entries will be judged on programming ability, graphics expertise, and musical content as well as overall presentation. All entries must be previously unreleased material. Entries should be accompanied by contact details, and the winner will be featured in Binary Zone P.D. For further information, or to enter the contest, contact:

   Jason 'Kenz' Mackenzie
   Binary Zone P.D.
   34 Portland Road
   Droitwich
   Worcestershire
   England
   WR9 7QW

@(A): Possible Products for SuperCPU 'Puter

PROTOVISION, a game development company, is currently working on compression tools for the SuperCPU, as well as some utilities for the new unit that are designed to take full advantage of the 16 bit processing power of the 65C816. In addition, the company is investigating the possibility of developing a new graphical operating system for the unit that will run in 16 bit mode and take advantage of the 16 megabytes of addressing and the new opcodes available in the accelerator. The new system will be able to multitask and offer several graphics modes and capabilities.

@(A): New OS Version Available

For those who enjoyed reading about Andre Fachat's OS/A65 operating

system in the last issue of Commodore Hacking (Issue 13, Section: os), Andre has updated his multitasking OS to version 1.3.10b. New in this version is:

* MORE AVAILABLE TASKS and STACK SPACE for systems w/o MMU (C64): New compile option STACKCOPY for systems without MMU. Allows more tasks and gives each task more stack space.

* new 9600 baud RS232 driver for C64! (see comp.sys.cbm FAQ)

* new 6551 ACIA driver for the C64 (connected to IRQ line)

* new 16550A UART driver for all supported machines!

* New GETB and PUTB kernel calls, to get and put a whole data block from and to a stream. It is used by the FSIEC filesystem currently.

* Better NMI handling:
  New CTRLNMI kernel call, and modified SETNMI call. SETNMI now sets the NMI routine address, and enables NMI routine chaining. There also must be a link to a control routine that can enable and disable the NMI line. Therefore FSIEC and FSIBM now call CTRLNMI to switch the NMI on and off around their time critical regions.

* Introduced return code to device IRQ routines. This allows to return from the IRQ routines prematurely, if one IRQ routine signals that it has removed an IRQ source (if SYSINPORT is not available).

It is available on the World Wide Web at:

   http://www.tu-chemnitz.de/~fachat/csa/

@(A): Current Releases for the Commodore from CWI

Computer Workshops, Inc., is currently distributing NewView, an image effects generato, and HyperLink, a hypermedia authoring system. More information on these titles and their upcoming 3D Game, "Nether", are available at:

   http://www.armory.com/~spectre/

@(A): Commodore Hacking Selected for Inclusion in PC Webopaedia

Sandy Bay Software, Incorporated has selected Commodore Hacking's World Root WWW Site to be included in a virtual encyclopedia of WWW sites. Visit the PC Webopaedia at:
Dear Webmaster:

   http://www.sandybay.com/pc-web

@(A): LOADSTAR's Pass Around Issue is Available

J and F Publishing has announced that LOADSTAR Issue #148 has been selected to be a "pass-around" issue. This issue is available for free and is intended to allow non-subscribers a chance to see what is in the disk-based monthly magazine. Issue #148 is available at:

   http://www.loadstar.com/pass.html

Wraptorized and ARCed versions are available for 1541 and 1581, while Compression Kit, .d64 images, and PKZip version are available in 1541 format.

Screen shots of the issue are available at:

   http://www.loadstar.com/148contents.html

The issue is packed, filling over 700 kB, and includes articles like:

* How to Copy Files
* Super Snapshot Bypass Switch Installation
* 'Lectronic Formulator
* Directomeister II Directory Editor
* Menu Toolbox III, as published in C=Hacking #14 (Reference: toolbox)
* GEOS Fonts (Ronda and Triana)

Download the issue today, and share it with your friends and user groups.

@(A): Hey! It's the Commodore Man!

If you're in the market for some used equipment or require some service

on your CBM system, contact:

   Jon Searle, The Commodore Man
   Service and Software
   1307 Golfview Drive
   Grain Valley, MO 64029

Jon offers over 1000 titles, documentation, books, magazines, and hardware. A catalog can be requested for a nominal fee. Until December 31, 1996, Jon is offering a "Buy 3, Get 1 Free" offer on software titles. Write for details and restrictions.

@(A): GEOS III, Revenge of the 8-bit GUI!

Maurice Randall, creator of such GEOS offerings as GeoFAX and many utilities for CMD, has announced that he is formally working on GEOS 3.0. He has successfully reverse engineered GEOS 2.0 and is now working to incorporate changes into the OS that will provide more seamless support for peripherals announced since the release of GEOS 2.0. Among the changes is a number of bug fixes to the original GEOS OS code and some enhancements to allow shortening of the OS code or faster execution. A time or formal name for the project has not been determined, but the new version will require some type of RAM expansion.

At a recent Lansing Area Commodore Club meeting, Maurice demonstrated some of the changes that may show up in the final GEOS 3.0 system. They include:

o Removal of 15 file limit in file lists.
o Ability to use CMD devices in Native Mode.
o Ability to read CMD FD DD disks in a 1581 drive.
o Ability to create CMD Native partitions on a RAMDisk.
o Ability to read/write MS-DOS floppies as native files on 1571,81, or FD.
o Ability to read single bytes from drives.
o Ability to utilize 4 separate disk drives simultaneously

These changes are incorporated in new disk driver code that can be used by any existing GEOS application. The new drivers utilize a radically different Configure program with more options than the current setup application of the same name.

Maurice stated that he will likely take a half-written desktop replacement he has written titled "Dashboard" and develop it into what will become the replacement for DeskTop in GEOS 2.0.

Although the new system will require the use of RAM expansion, the system will not require a SuperCPU or other speed enhancement unit to operate.

@(A): Explosive Commodore Surfing Power

Brain Innovations, Inc., recently announced that they had revamped the popular WWW Links pages on Jim Brain's Commodore home Page. The new site, completely automated, offers many advantages over the older set of pages. The new site, called CaBooM!, can display links in either HTML TABLE or UNORDERED LIST format, include or exclude graphics, and include or exclude link descriptions. The site is organized into categories and sub-categories. Surfers can automatically add new sites to the system and specify which categories fit the site. Users can also update sites at a later date by specifying their user ID and password used to create the site. New and updated entries are identified with appropriate comments, and sites can be listed in multiple categories. Check out the new system at:

   http://www.msen.com/~brain/cbmlinks/index.html

==========

@(#)trick: HEAVY MATH - Part 1: Introduction to Linear Programming (LP)
           by Alan Jones (alan.jones@qcs.org)

This article describes the Linear Programming problem. LP software is being developed for the C64 to emphasize that the C64 is capable of solving HEAVY MATH problems. It describes some of the C64 program design issues and gives readers an opportunity it influence the development of the software.

@(A): Introduction

Linear Programming, LP, is the simplest type of constrained numerical optimization problem. It's simplest (standard) form is:

```
     max P = c'x

     s.t. Ax  = b
           x >= 0
```

That is, we want to find the solution vector x which maximizes the function P, which is a linear combination of elements of x, while satisfying the linear equation Ax = b, and with each element of x >= 0. A is a matrix with typically more columns than rows. The maximization problem itself is easy, except for the inequalities and determining which elements of x are fixed at a bound and which are free.

Assume x is a vector of 20 variables and we have 10 equality constraints making A a 10 by 20 matrix. Solving the problem involves choosing 10 of the variables to be bound (at zero), eliminating those 10 columns from A and solving the reduced linear system, say B*xb = b, for the other 10 elements of x, xb. Then sort through the solutions to find the feasible solutions that satisfy all of the constraints, x >= 0, and chose from them the solution that will maximize the function P. Taking 20 columns of A 10 at a time we will have 184,756 solutions to solve and evaluate!

@(A): LP Development

LP is an important problem that requires a computer and clever algorithms to solve non-trivial problems. The development of LP algorithms parallels the early development of computers. George Dantzig published his first paper on his simplex method in 1947. LP problems have long been studied in mathematics, economics, and business fields, but the simplex method and the computer were the big breakthroughs. Just imagine the problem of allocating and distributing limited resources to millions of soldiers in WW II. There are many types of problems that can be solved as LP problems, and some have specialized algorithms for their efficient solution.

@(A): LP Examples

The inequality constraint occurs naturally. Many processes are irreversible. For example you can burn fuel in rocket at a positive fuel flow rate to produce thrust, but you can't unburn the exhaust to refill the tanks. A table leg can push on the floor but not pull (unless bolted). A rope, cable, or chain can pull but not push. You can't buy or sell negative quantities of a new item. You can't start assembling a machine before the parts arrive.

LP problems can also be written in more general forms. Perhaps the most general is:

```
    max P = c'x

    s.t. bl <= Ax <= bu
         xl <= x  <= xu
```

Where bl an xl are vectors of lower bounds and bu and xu are vectors of upper bounds.

The LP forms can be manipulated with simple algebraic operations and by adding constraints and "slack" variables. Most numerical optimization problems are set up to minimize a cost function. Historically, LP problems are set up as maximization problems. If your function P is a total cost to be minimized, simply multiply the vector c by -1 and maximize instead. Multiplying the i'th constraint equation by -1 will change the sign of all the elements of the i'th row of A, bl, bu, and change the direction of the inequalities, but not change x(i). The i'th constraint above is actually two equations and can be written (neglecting the subscripts):

```
     Ax <= bu
     Ax >= bl
```

To convert them to equality constraints:

```
     Ax +  y1 = bu
     Ax + -y2 = bl
```

y1 and y2 are additional variables both >= 0 added to take up the "slack" in the inequality equations. When the constraint is free (i.e. could be ignored) the slack variable is >0. When the constraint is active, the slack variable is constrained to zero. The slack variables are simply appended to the vector x and treated as normal variables.

Lower bounds on x can be removed with a change of variable, x = y - xl, without adding more rows or variables.

```
xl <= x <= xu,  becomes  0 <= y <= xu - xl
```

Upper bounds on x can be handled by adding slack variables. The expanded problem could look like:

```
Ax = b
x + xs = xu
x >= 0, xs >= 0
```

This can double the length of the x vector and add an equal number of equality constraints. A fair amount of algebraic manipulation is possible to get a LP into the form required by the solution software. This can significantly increase the problem size. Good software will use the more general forms of the LP program and use a more sophisticated solution logic instead of increasing the problem size.

Mixture problems are probably the easiest to describe for illustrating the importance of LP. Suppose you are producing cans of mixed nuts. You want to find the percentage of each nut to mix and package at the lowest cost, or maximum profit. You check the commodities prices of the various nuts available at different times and places. c(i) could be the price per pound of each nut, i, and x(i) would be the percent of that nut to be mixed. One constraint is that the percentages total 100. The Marketing Department may demand no more than 25% peanuts and no less than 5% of Cashews, pecans, and almonds. Each nut may be scored for crunchiness, and other aspects of taste to meet other constraints. Of course no x(i) can be less than zero. You can do the same thing with "real fruit juice" drinks. Mix the cheapest available fruit juices while balancing sweetness, acidity, color, taste, and other factors.

Another type of problem is project scheduling. Suppose you are going to design and build a bridge, or spacecraft. You identify all of the tasks that must be accomplished, and then estimate their completion time and constraints. E.g. task 43 can't begin until tasks 16, 17, and 41 have been completed. You could then set up a LP problem to schedule each task so as to complete the project in the minimum time. Various other performance criteria can be embedded in a LP problem.

@(A): Non Linear Programming (NLP)

LP is also a special case of Non Linear Programming, NLP. The NLP problem is similar to the LP problem except that the objective function P and constraint equations can be nonlinear functions. NLP is a much more expensive problem to solve. An approach to solving a NLP is to linearize the problem at a point to get a LP subproblem. Solving the LP gives a search direction for the NLP problem. You then advance in the search direction until you have made sufficient improvement in the objective function or diverged too far from the constraints. You then correct back to the constraints and linearize again. The sequence is repeated until the solution is found.

@(A): C64 Software?

Computers were not created by secretaries and typesetters. Although computers can do a lot of things, they were created to do Heavy Math. The first electronic computer, the Attanasoff-Berry Computer, ABC, was built to solve systems of up to 29 linear equations, although it could be adapted to solve other problems as well. The second and third, ENIAC and EDVAC, were intended to solve ODEs (Ordinary Differential Equations, i.e. computing artillery range tables), as well as more general usage.(ENIAC's first operational use was solving a 25 by 25 set of linear equations from Los Alamos to see if an atomic bomb might be feasible. The ABC could have solved this problem, but it had a read/write reliability problem and Attanasoff was called up for other war related work instead of perfecting the ABC.) And I have already discussed the LP problem.

Software for the C64 has been readily available and published in the C64 related literature for solving systems of linear equations and numerically integrating differential equations (4th order Runge-Kutta being typical). However nothing is available or published for solving LP problems on a C64. I do recall seeing a one inch ad in some Commodore magazines trying to sell software to solve small LP problems on a C64, maybe 15-20 equations, but I never read any reviews of it and I always suspected it to be poor software. I intend to plug this hole by writing a good LP program for the C64.

This is Part 1 of at least a two part series on LP for the C64. Part 2 will include the presentation of the C64 LP program. I usually hate

multi-part articles that could have been published as one large article. However, in this case Part 2, and indeed the C64 LP computer program, has not been written yet. Part 2 will be dependent on reader feedback! No feedback means no reader or user interest and I won't bother to submit Part 2. I have not written a LP program before, or even used one much on other systems. There is a very good chance that some readers will have experience with LP software and can offer practical suggestions for writing C64 LP software. I would also like to hear from any reader who may have some type of LP problem that they would like to be able to solve, perhaps something related to a hobby or home computing application. Also, someone may be able to point me to an ideal LP reference book or even provide suitable source code to examine.

Choosing a LP solution method for the C64 is not an easy task. Appropriate reference books and papers are hard to come by. There is nothing in the CBM literature that I am aware of. Many books have been published on the LP subject. However, most are old books written for business students or for training clerks to solve small problems by hand, and they use outdated methods. These are easily found in local libraries and at small colleges. LP is still an active research area and many new journal articles are still being written, as well as new books. However, the focus in on solving VERY large LP problems using sparse matrix algebra techniques, and newer methods such as Karmarker or interior-point methods. There is even a LP FAQ available. Two of the best references that I have found so far are: "Computer Solution of Linear Programs", J. L. Nazareth (1987), and "Advanced Linear Programming", B. A. Murtagh (1981). The LP source codes that I have looked at were either too crude, suitable only for small problems, too poorly documented, or suited only for large problems on large computers. This gives you some idea of where I am at, should you want to provide helpful feedback.

Our C64 (and 128) does arithmetic slowly, but reliably, and has limited memory. We would not want to solve a large LP problem with thousands of constraints and variables on a C64 even if we could. Still, we want to be able solve problems as large as practical using efficient methods. At the heart of the solution algorithm we must be able to solve an n by n system of equations, where n is the number of constraint equations. For, say n = 70, we will need 24,500 bytes to store the full matrix, leaving the rest for the OS/language/LP program. The simplex type solutions typically require computation time = K * n~3, so we will not likely want to solve any LP problems on the C64 with more than 70 constraints.

The desire to solve huge LP problems also motivated the development of sparse matrix algebra techniques. Matrices associated with LP tend be very sparse. The idea is to be able to store only the non-zero elements, to store them in data structures that can be used effectively by linear equation solvers, and to use larger but often slower external memory efficiently. These codes have a higher overhead in terms of code size and computation/FP multiply. However, these codes can compute the same results with substantially fewer multiplies than simpler code for dense or full matrices. They often perform faster overall, at least on scalar computers. Our 6502 can zip through complicated data structures and ML code fairly fast compared to the cost of a FP multiply. Using sparse matrix techniques is possible with the C64, but not required. I think the complications of using sparse matrix methods is not warranted in this case. Users wanting to solve large sparse problems are free to disagree.

@(A): Types of LP Problems

There are several LP solution methods: (primal) simplex, revised simplex, dual simplex, primal-dual simplex, Karmarker, and others. There are also many variations among these. I have chosen the revised simplex method for the C64 program. Some other choices might be better for some types of LP problems, and I may also write a primal-dual simplex program.

In the revised simplex method, the constraint matrix A can be stored outside of main memory (on disk or REU) and brought in a column at a time. The matrix A can be stored in a sparse matrix form. This is probably a good idea for our slow disk storage, but unnecessary for our REU storage. A square matrix B will be based on a set of n columns of A. At each iteration a new column of A is chosen to enter B and an old column is re moved. Each column represents an element of x and the objective is to get the unbound variables selected into B with the remaining variables set to their limits. At each iteration we have to solve many linear systems using B with different right hand side vectors and then update B to account for the column addition and removal.

There are several ways to work with the matrix B. B can be explicitly inverted. The inverse can then be explicitly updated using the Sherman-Morrison-Woodburry formula, or updated in product form by storing a sparse matrix factor that accounts for the update. The later is

preferred for sparse implementations, but the storage used grows with each iteration and eventually a new explicit inverse of B will have to be computed. Both methods are numerically unstable and can have excessive growth of roundoff errors, necessitating error checks and occasional reinversion.

B can also be efficiently used in an LU factored form. B = LU where L is lower triangular and U is upper triangular. The updating is difficult, especially when exploiting sparcity and external storage. Bartels and Golub (1969) developed a numerically stable way of updating the L and U factors that enabled L to be stored externally, with U stored in main memory. Forrest and Tomlin (1972) developed a method that allows both L and U to use external storage. This is the method used in most commercial codes for solving large LP problems. However, it is not numerically stable and needs to be monitored and refactored. Sanders (1976) developed a variation of the Bartels-Golub method that is stable and allows most of U to be stored externally. Fletcher and Matthews (1984) developed a stable method for updating the LU factors explicitly in main memory.

In the Fletcher-Matthews update we have: PBQ = LU. B is the unfactored matrix, P is a row permutation matrix, Q is a column permutation matrix, L is a unit lower triangular matrix, and U is an upper triangular matrix. L and U are stored explicitly (not in a factored form) in a square matrix B. L and U are first formed using a normal LU factorization (Gaussian elimination). P is a row permutation matrix that represents the partial pivoting normally used to stabilize Gaussian elimination. Q is an optional column permutation that represents the full pivoting that could be done with Gaussian elimination to get better numerical stability. P and Q are completely defined by integer arrays. In the update, one column of B is removed, the rest of the vectors are shifted left to fill the gap, and the entering vector placed in the right most column of B. L, U, and P are then updated to a stable factorization of the new B matrix. Pivoting is still required for stability, but only two adjacent rows can be exchanged. This is a weaker form of stability than Gaussian elimination with partial pivoting. It may be advisable to use some more extreme measures to assure greater accuracy, such as preliminary scaling of the problem (equilibration), full pivoting in the initial factorization, or doing a final refactorization. Q is not changed during the update (no column pivoting for stability), but we will have to keep track of where each column of A is located in B (or the LU factorization of B). The update is also cheaper when the column of B removed is farther to the right.

I have chosen to use the Fletcher-Matthews update. This choice is ideal for dense LP problems, but less so for sparse problems. It limits the number of constraints that we can handle. It also has consequences for other program design choices that must be made for the C64 LP program. Using the simple standard form keeps the program logic simple, but converting problems with inequality constraints and upper and lower bounds to standard form will make the problem size bigger. This will be more expensive to solve when not taking advantage of sparcity, and make our size limit more significant. I am leaning toward solving LP problems in the form:

```
    Max P = c'x

    s.t.   Ax = b
           xl <= x <= xu
```

There are many more program design choices to make. There are different strategies for selecting the entering and leaving variables (columns). There are different ways to pick a starting B matrix. There are different ways to perform "Phase 1" of the revised simplex algorithm. Many of the internal tests of floating point results involve tolerances. These tolerances do affect the performance of the program. However, I have only seen arbitrary suggested tolerances. These tolerances should be determined by the floating point precision used, the problems size, matrix condition number, and perhaps some problem specific parameters. I have not seen any numerical analyses indicating how to set the tolerances.

I have completed the subroutines to perform the Fletcher-Matthews update, matrix factorization, and solving linear systems. Your interest and feedback can influence other major parts of the program, or even convince me to use a different updating method. Or perhaps everyone would rather not see the sequel? You can reach me via e-mail at: alan.jones@qcs.org

Since I am writing this software it will be developed in the COMAL 2.0 language, which is fast and very readable (and thus easily translated to other languages). A typical commercial LP code represents about 10 man

years of development. The C64 software will represent perhaps 10 days, plus past research, or 10 weeks of part time effort, and be submitted for the next issue of C=Hacking. It won't be the best software, but it will be good. Over a period of perhaps a year, we can incorporate and publish changes (hopefully not actual bugs). Then it can be translated be to ACE assembly or some other language to make it available to more C64/128 users.

@(A): Conclusion

The C64 LP software is being developed simply to fill an empty slot in C64 software. It is also to emphasize that the C64 can indeed be used for HEAVY MATH, subject only to constraints of limited speed, memory, and software. There is little demand for solving LP problems on a C64, otherwise it would have been done many times already. This article will not interest many C64/128 users. It does not even come close to describing how to write a working LP program. It does describe what LP is and some of the issues involved in designing a LP program for the C64. I do thank those that read this far. Don't forget to write.

==========

@(#)mags: Hacking the Mags

Not everything good and/or technical comes from Commodore Hacking, which is as it should be. (We still think we have the most, though...) Thus, let's spotlight some good and/or technical reading from the other Commodore publications.

If you know of a magazine that you would like to see summarized here, let C=Hacking know about it. These summaries are only limited by Commodore Hacking's inability to purchase subscriptions to all the Commodore publications available. We are very grateful to those publications that send complimentary copies of their publications for review.

@(A): Commodore World (http://www.the-spa.com/cmd/cwhome.html)
   Although a bit late, Issue 15 made its way to our mailbox and opened with an apology from VP Charles Christianson detailing why the current issue took so long to reach subscribers. As we suspected, it was due to SuperCPU shipments and general bug-swatting. However, another factor that wasn't as obvious was some personnel changes in the publication. Gaelyne Gasson goes over the various graphics formats and how to convert them to viewable formats on the Commodore. The GEOS programmers will appreciate Maurice Randall's article on VLIR files, and Doug Cotton presents Part 2 of his discussion on file transfer utilities. The demo scene gets a little press with Sherry Freedline's piece on demos, including a section on _Driven_, reviewed below. Commodore Hacking even got a mention or two in Max Cottrell's report on the Lansing Area Commodore Club Expo '96.

@(A): DisC=overy (http://www.eskimo.com/~drray/discovery.html)
   Arriving on the Internet October 1st, Issue 2 of this new publication maintains the level of content started in Issue 1. We suppose it's too early to tell, but it looks like one will show up every 4 months. The issue starts with three detailed pieces on VIC video techniques, including a discussion of the Super Hires FLI technique by Roland Toegal and 'Count Zero', a starter article on using raster interrupts by Mike Gordillo and 'Dokken', and how to use simple text scroll routines in programs by Mike Gordillo. Steve Judd details the basics of using the SID chip, and Andreas Varga steals the issue with an interview with SID creator Bob Yannes. The article is a must read. The highlights of the hardware section is a Atari 2600 cartridge reader for the VIC-20 by Ravid Noam and how to upgrade a Commodore 16 to 64 kB by Martin Gierich.

@(A): Driven (http://soho.ios.com/~coolhnd/)
   In addition to the funky banner on Driven #15, the issue mentions the resurgence of the Demo Scene and congratulates all the recent Driven 4kB Competition entrants. If you are interested in demos and what effects are possible, be sure to check out the recent entries. #15 details the CMD Swiftlink in an article by Perry Eidelbus. Users undecided on purchasing a SL, or programmers unsure of whether to develop for the unit should read this piece. In a look back, 'The Hobbitt' runs through the history of the NTSC demo scene. Also, if you didn't get a chance to take a look at DisC=overy yet, there's an interview with editor Mike Gordillo in this issue.

   In between #15 and #16, Driven published the "Driven 4K Compo Edition", containing reviews and comments on the entries submitted for the recent Driven competition.

Driven #16 contains a look into the world of Computer Workshops, Inc. (CWI) by Cameron Kaiser, as well as a detailed description of Craig Bruce's ACE OS, available on the Internet. As well, there are the usual notices of new demo releases and groups. The editors remark that they feel #16 is the best Driven yet.

@(A): LOADSTAR (http://www.loadstar.com)
Sometimes, we read LOADSTAR purely for the entertainment factor. And I don't mean the games on the disk. In Issue 145, Jeff Jones outlines his top ten email pet peeves. #145 delves into a rare topic in C64 circles: Musical Instrument Digital Interface (MIDI). Fender goes over the protocol and how it can be used with a 64/128. On the heels of that article is a piece by John Serafino that creates MIDI tracks for your own enjoyment. Bo Zimmerman presents a "Presenter" for LOADSTAR that utilizes GEOS. In addition, Bo gives GEOS users a handy utility that archives files and can even create .d64 images. To supplement Maurice Randall's _Commodore World_ article on VLIR files, Roger Detaille details the revisions in the directory that GEOS disks dictate. As a bonus, Issue #145 contains Driven 12, reviewed in C=Hacking Issue 13 (Reference: mags) as well as DisC=overy #1, also reviewed last time.

Issue 146 starts off on an apologetic note, as Jeff repents for redistributing DisC=overy #1 not in its entirety. It brings up an important point that compilations like DisC=overy and C=Hacking can only be freely redistributed in their entirety. Diving into the issue, developers looking to design their own fonts might find use in Anthony Rose's Font Studio, Bob Markland's Font Viewer II, or Bob's Font Studio Printer. Jeff presents his entry in the Directory Editor arena: DirectoMeister I.

Demo Scene folks will enjoy "Omni's First Demo" on Issue 147. Continuing with the video theme, Andrew Martin present Hires Sketch II, an art creation application. Of interest to GEOS DTP (Desktop Publishing) folks is a set of two GEOPaint documents containing some clip-art.

As mentioned in Newsfront (Reference: news), Issue 148 has been released to the Commodore community as a "pass-around" issue. Distribution is encouraged. Although not technical in nature, everyone should read the piece describing this issue. It gives some insight into the workings of LOADSTAR and its ex-parent company, SOFTDISK. Fender Tucker shows how to add a bypass switch to a Super Snapshot cartridge, while Jeff revamps his Directomeister I into version II. As well, Menu Toolbox III, included in this issue (Reference: toolbox) is available on LS148. As well, Commodore Hacking #13 is included on the 3.5" disk version.

@(A): The Underground
Issue #14 is the last issue for this publication. It has merged with LOADSTAR LETTER. See Newsfront (Reference: news) for more information. We were hoping to get the last issue in to review it, but C=Hacking's recent relocation sent the issue off into never-never land. Anyway, we wish Scott Eggleston and his family the best. He will continue to edit LOADSTAR LETTER.

Other magazines not covered in this rundown include:

* _64'er_
* _Atta Bitar_ (_8 bitter_)
* _COIN!_
o _Commodore 64/128 Power User Newsletter (CPU)
o _COMMODORE CEE_
o _Commodore Gazette_
* _Commodore Network_
* _Commodore Zone_
o _LOADSTAR 128_
o _LOADSTAR LETTER_ (We received an electronic copy, but couldn't print it)
* _Gatekeeper_
o _Vision_

Notes on Legend:
* = We have never received an issue of this publication.
o = We have not received a new issue of this publication to review.
\+ = We will begin reviewing this magazine in the next issue.

In addition, others exist that C=Hacking is simply not aware of. As soon as we can snag a copy of any of these, or get the foreign language ones in English :-), we will give you the scoop on them.

==========

@(#)net: The Commodore Telnet BBS

by Bo Zimmerman (ez13942@swt.edu)

@(A): Overview

The following are instructions for setting up a Commodore computer as a telnet-able BBS. It relies on a modem connection with a PC running LINUX with a telnet daemon. The Commodore is connected to the PC via a null modem cable. The LINUX box has a modified version of minicom, which comes with the slackware distribution (and others I would imagine), and a shell script, all described in greater detail below. The essential goal is this: when a user telnets to the LINUX machine and logs in as the BBS user, the PC will run the modified minicom, which is set up to communicate with the COM port connected to the Commodore. When minicom starts up, it will signal the Commodore that a connection has been made by setting the modem port's DTR signal. The Commodore BBS program goes online because the DTR line is attached to the DCD line in the cable. So long as the user is still in minicom, the connection remains, and wa-la! A Commodore on the net!

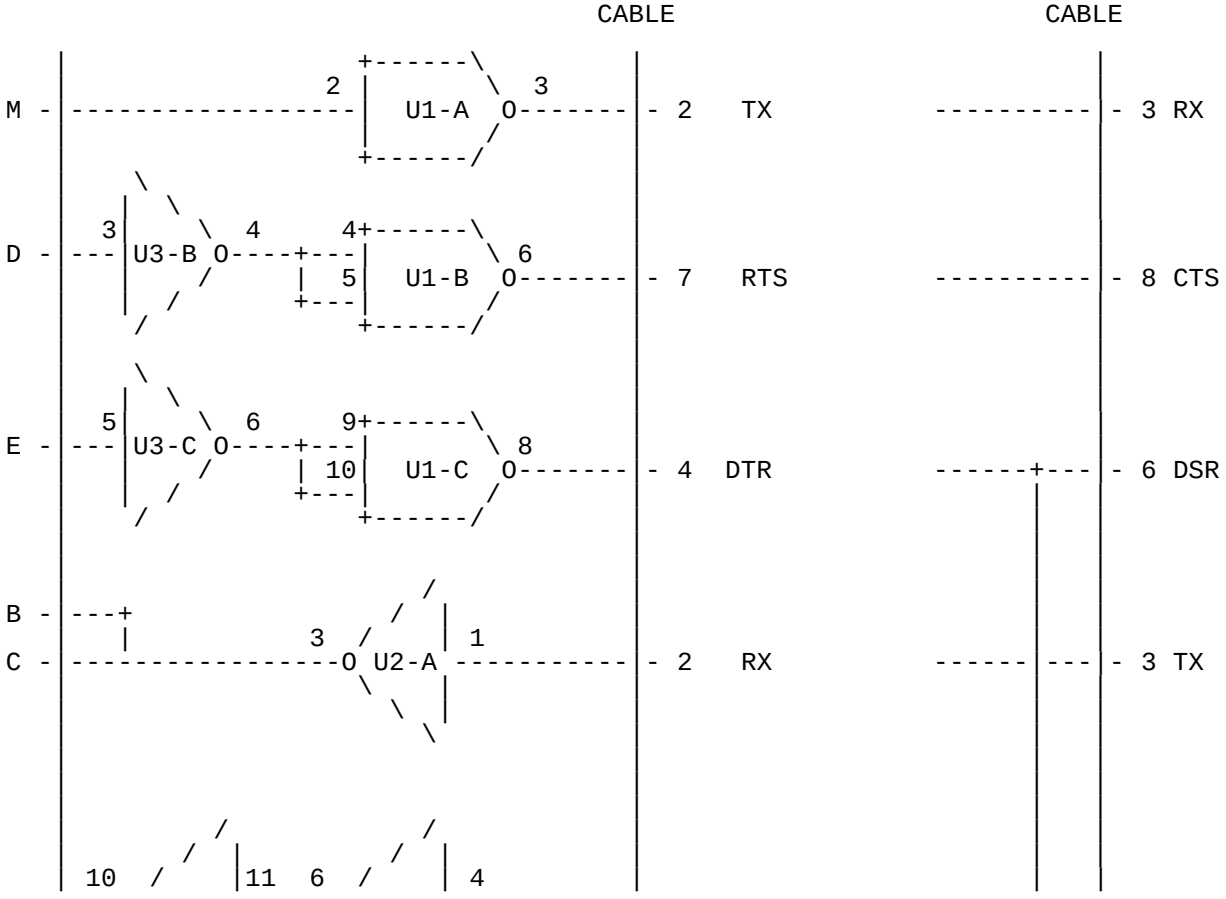
Part I : Required Components (SubRef: 1)
Part II : RS232 Adapter Instructions (SubRef: 2)
Part III: Setting up the LINUX box (SubRef: 3)
Part IV : Setting up the Commodore BBS program (SubRef: 4)
Part V : What's missing (SubRef: 5)
Part VI : Credits (SubRef: 6)

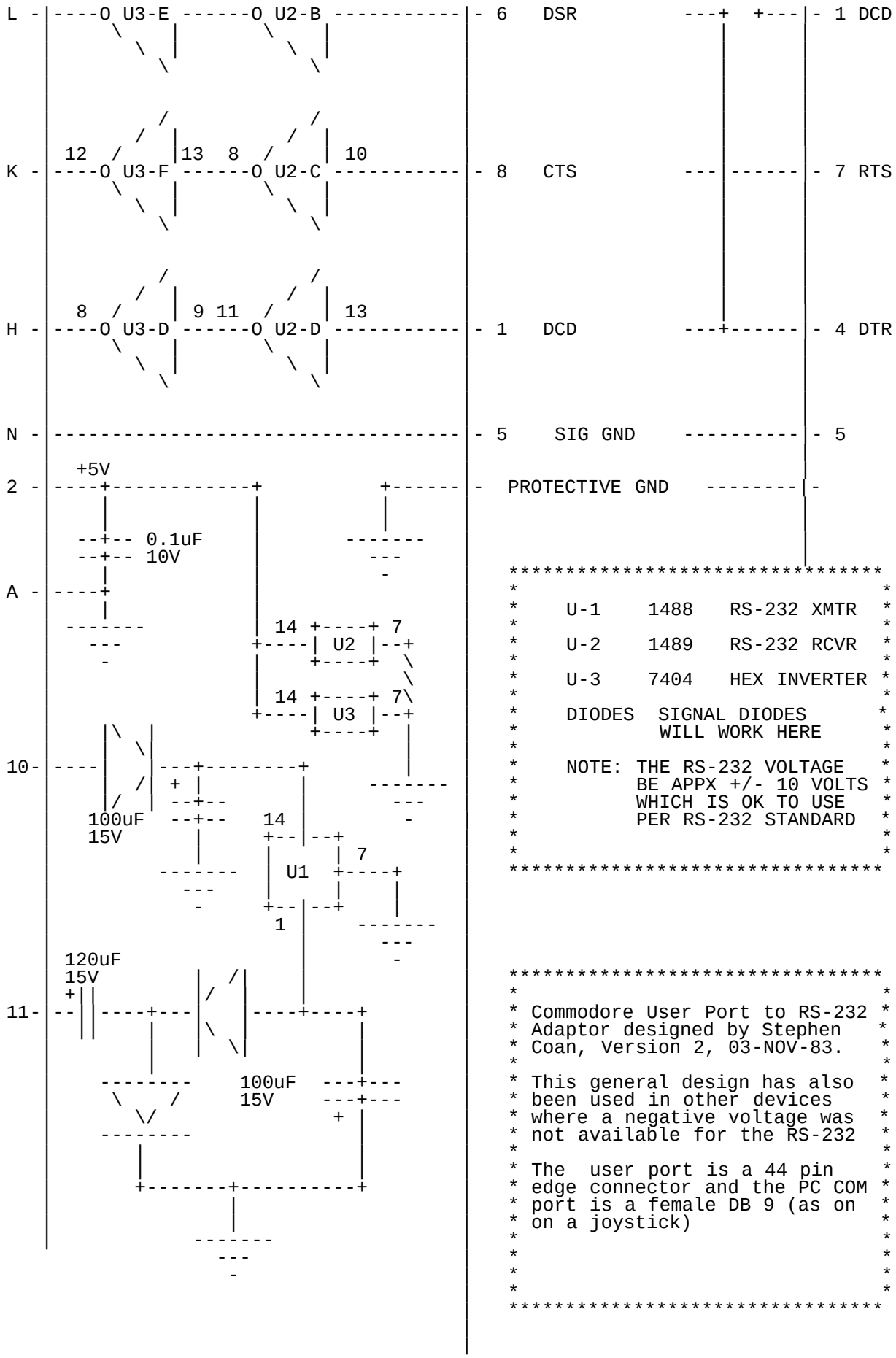
@(A)1: Part I: Required Components

1) Commodore 64/128/VIC-20
2) Sufficient Commodore drives for your BBS software.
3) Commodore BBS software with DCD initiation capabilities (see part IV)
4) Standard RS232 null modem cable adaptor for the Commodore
5) PC 386 or better running Linux
6) Network capabilities for the Linux machine (ethernet card, PPP connection, or other connection)

@(A)2: Part II: RS232 Adaptor Instructions

(This is cut and pasted from some instructions I downloaded off of ftp.funet.fi and then modified for a standard 9 pin female COM port on a PC).





You'll want to follow the instructions for the NULL modem cable for the sake of this project, so use the pin settings on the right hand side.

@(A)3: Part III: Setting up the LINUX box

You should know now that if you don't have root access to the LINUX machine, you should give up now. Secondly, you need to have the machine already configured with respect to its network hardware and the telnet daemon. The popular slackware distribution sets all this up during its installation process.

The first step will be to create the above shell "new_minicom" and its accompanying configuration "cua0".

1. Log on as root, and enter the /usr/bin directory.
2. Run minicom by entering its name followed by the name of the port you'll be using. For instance, enter:

   "minicom -l -o cua0" for com port 1.
   "minicom -l -o cua1" for com port 2.

3. Enter CTRL-A followed by 'z' to view a menu of options.
4. Enter CTRL-A,P for communication parameters. You'll want the parameters to be: 2400 baud, 8 bits, no parity, 1 stop bit. Exit this menu when done.
5. Enter CTRL-A,T for terminal parameters. ANSI, Backspace, and enabled are fine settings.
6. Enter CTRL-A,O for configuration parameters. Select the third option-communication port setup. Make sure the serial device reads "/dev/cua0" for com port 1 or "/dev/cua1" for com port 2. Baud/parity/bits should read as you set them above. Hardware and software flow control should BOTH be turned OFF!
7. Exit back to the parameters menu and select "modem and dialing." The reset and initialization strings should be blank. Auto baud detect, hang-up drops DTR, and modem has DCD line should all read YES.
8. Exit back to the parameters menu again and hit the option "save configuration as cua0" (or cua1 if you are using com 2).
9. Exit minicom altogether by entering CTRL-A,X.
10. We will make this our TEMPORARY new_minicom by entering from the shell:

    cp minicom new_minicom

If you want to test the RS232 cable, now would be a good time. Run minicom with the following options:

   new_minicom -l -o cua0   (or cua1 for com port 2)

The next step is to create the BBS "user."

1. Log on as root and run the program "adduser." Call your account "bbs" and give it the name "bbs." After the user is created, you should have a directory called "/home/bbs."

2. Now load the file "/etc/passwd" into emacs and find the line containing the information about your BBS. The characters between the first and second colons in the line are the encrypted password. Delete these characters. Next change the default shell (the part following the last colon) to point to the file /usr/bin/new_minicom -l -o cua0.

3. When complete, excepting the ID number (504), your line should look identical to this one:

   bbs::504:100:bbs:/home/bbs:/usr/bin/new_minicom -l -o cua0

   If you are using com 2, change the shell command to read "cua1" instead of cua0.

The next step is a little tricky. You'll need to make some changes to minicom to make it much more secure against abuse by users. Without changes, the user can enter all the commands you did above! It will be necessary to have extensive knowledge of C and knowledge of compiling in LINUX to perform these changes. ;)

Now that you're frightened, you'll be glad to know that the necessary files are available at:

   147.26.162.107

in the "lib" subdirectory.

The only files you'll really need are "new_minicom", which must be copied to your /usr/bin directory, and the file "wb.rc" (discussed below), which

must be copied to /home/bbs. Also available is the file "newminicom.zip" which contains all the modified source code.

The following changes, to the best of my memory, were made:

1. All CTRL-A commands have been disabled, with the exception of CTRL-A,X.
2. Minicom will automatically exit on the reception of the sequence CTRL-A,CTRL-B,CTRL-C,CTRL-D from the modem.
3. Spawning out has been disabled.
4. The status windows have been removed.
5. ANSI has been made permanent.
6. A "busy" message for already connected processes has been added.
7. A "cleaning up" message for the com port has been added in the event of an abnormal exit.

After new_minicom is set up in the /usr/bin directory, you may think you're done. Well, you almost are! You may need to make sure there are no protections on the com port, on minicom, or on other necessary files. Do this with the "chmod" command. You'll want to turn on read and write for the com port as follows:

```
chmod +r /dev/cau0     (or cua1)
chmod +w /dev/cau0     (or cua1)
```

Read and execute abilities can be added to minicom similarly:

```
chmod +x /usr/bin/new_minicom
```

Now you're done, right? No, but the last step requires a bit of explanation. Our telnet session is maintained because we have an actual user logged on and running a program. Should this user disconnect themselves from their telnet session without logging off, we are actually left with an open session of new_minicom still running in LINUX. Normally we wouldn't care, except that so long as new_minicom is running, the com port is protected from use, such that the system will be eternally busy! What we need to do then is to have a shell script, with root privileges, that will keep an eye on copies of new_minicom running without a telnet daemon connection. The following shell script will do that.

>From the /home/bbs directory, create the file "wb.rc" (I call it the watch-boy) and enter the following lines:

```
set temp=1
while (temp=1)
do
   pid=`ps -auxg | grep 'new_minicom' | grep -v 'grep' | grep 'bbs' \
   | grep '?'| awk '{print $2}'`
   tty=`ps -auxg | grep 'new_minicom' | grep -v 'grep' | grep 'bbs' \
   | grep '?' | awk '{print $7}' | grep '?'`
   if [ 'expr $tty : "?"' ]
   then
     kill -9 $pid
   fi
   sleep 60
done
```

[!NOTE!: The two lines with the backslash (\) are extensions of the prior lines. For instance, the characters "| awk '{print $2}'`" should immediately follow the end of the line that reads "...grep '?'". Do not include the backslashes!]

What this actually does is to look for occurrences of "new_minicom" being run by a user called "bbs" in which there is no terminal connection. It then kills those processes and goes to sleep for awhile before checking again.

This file can also be downloaded from 147.26.162.107 in the "lib" subdirectory.

Last thing to do then is to activate the watchboy by adding the following line to the file /etc/rc.d/rc.local /home/bbs/wb.rc &

You can start up the watch boy as root without rebooting the machine, but the above will make sure it is started up whenever the LINUX box is restarted.

Now on to the Commodore side.

@(A)4: Part IV: Setting up the Commodore BBS program

Thankfully, doing this is MUCH easier than setting up the LINUX box. The requirements on the Commodore end are actually very few, since what we end up doing is have the LINUX machine act as our modem and phone connection. As far as the BBS program is concerned (with few exceptions), it is acting completely as normal.

You'll first want to select the BBS program to work with. Make sure it is one that is written in BASIC so that you can modify it easily. The BBS should also support ASCII, and preferably ANSI as well. For the time being, ANSI is the only way we'll get any color at all out of LINUX. The BBS program should support 2400 baud through the modem port. If not, you'll need to lower the set baud rate in minicom above.

Detecting a connection is accomplished by simply watching the Data Carrier Detect line on the modem port. That will be bit 4 in location 56577. When the following BASIC condition becomes true, the BBS should go online:

```
(peek(56577)and16)=16
```

The BBS program should go online by swinging its Data Terminal Ready signal high. This can be done with the following:
poke56577,6:poke56579,6

The BBS program should hang up whenever it detects the Data Carrier Detect line go low. That's done with something similar to the peek above:

```
(peek(56577)and16)=0
```

Lastly, the program will hang-up by doing two things. The first thing is to tell minicom to terminate. This is done by the following:
print#2,chr$(1);chr$(2);chr$(3);chr$(4);

Where print#2 is above you should substitute the proper modem channel for the number 2.

The second is to drop the Data Terminal ready signal by entering:

```
poke56577,0:poke56579,32
```

If the BBS program does all these things, it will perform admirably.

@(A)5: Part V: What's to Come

There are two current problems with this configuration for a telnet Commodore BBS. One is that Commodore graphics do not work. For a long time this baffled me, and I looked all over the minicom source for the solution. It wasn't there. Now I'm thinking it has something to do with the telnet daemon itself, which is the next place to check. When it's solved, I'll let you know.

The other problem concerns time lag. Normally, lag is unimportant, but some transfer protocols are now especially sensitive to lag time. For this reason, along with translation problems mentioned above and the sensitivity of minicom to its termination codes, make the operation of the stream transfer protocols on the telnet BBS impossible. Perhaps some modified uuencoding packet based protocol could be written and patched in to the BBS program. If any such solution presents itself, it will be pursued.

@(A)6: Part VI: Credits

Very few of these are my original ideas. Special thanks to Matt Beall of California (mathew@cinenet.net) for most of the modifications done to minicom. The project itself is the brain child of Henry Knoepfle of Arizona, who helped me through all the linux changes. Early in August, as soon as I get my own BBS program properly configured, I'll be putting it back up on the same machine that the ftp files are on. Telnet to it and log on as user "bbs" with no password. Good luck!

==========

@(#)usenet: UseNuggets

COMP.SYS.CBM: The breeding ground of programmers and users alike. Let's see what topics are showing up this month:

@(A): To C or Not To C, That Is The Question

Over the past few months, there has been some discussion of the C programming language, a common language used in the development of many UNIX, Windows, and Macintosh applications. We're not exactly sure what sparked the discussion, but GNU C came up very early. For those who don't know, the Free Software Foundation (FSF) is an organization that sponsors the creation and maintenance of many fine applications and utilities. One such group of applications are known as the GNU (GNU's Not UNIX) applications. Many people use the GNU utilities because they can be compiled and run on many different machines. One of the more popular apps is GNU C, which can be run on many computer systems as well as create object code for all the supported environments. The system is highly customizable, which explains why there was talk of both porting GNU C to the Commodore 64, and/or using a DOS or UNIX version of the compiler to create object code that would run on the C64. The thread has drug on for quite a while, but seems to be winding down. We're not sure there was a general consensus, but many remarked that GNU C is simply too large to run on a C64. In addition, the assumptions it makes about the supported hardware architectures makes cross-compilation a near impossibility. Still, many are searching for a way to bring ANSI C to the Commodore system.

Success may come from CMD, as Doug Cotton mentioned that they were looking into a 65C816 cross compiler with the hopes of porting a small free C compiler to the 64 environment.

@(A): The SuperCPU this and the SuperCPU that....

Now that the SuperCPU has started shipping from CMD, the newsgroup has been abuzz with questions and thoughts about the units. However, it seems the group is always one step ahead of CMD. Now that the units have started appearing on user's doorsteps, questions about the planned 128 unit and the SCPU RamCard have dominated the topics. The fact that the 64 unit actually contains 128 kB or RAM confused some folks, who wanted to know why they were paying for this extra RAM, and how they could take advantage of it now that they have it. Then, as fast as folks described that the extra RAM was actually used to "shadow" the ROM so that the KERNAL could run at full speed, the topic switched from present RAM to planned RAM.

Questions ranging from how much RAM would be available on the RamCard, to how fast the RAM could be accessed, to what style of modules could be used have been debated. CMD, the developers of the RamCard, acknowledged that the full address space of 16 megabytes would be available on the unit and that they would try to provide speeds as fast as economically possible. CMD has also stated that they will be utilizing the same 30 pin SIMM technology that is used in the company's RamLink product. Some Usenetters debated that 72 pin SIMMs were becoming more cost effective, but CMD countered that only those that upgrade to the full capacity of the unit would realize any cost savings.

The substantial increase in power the SuperCPU provides the programmer brought many questions and comments on planned or potential new operating systems for the unit. As of yet, none have surfaced, but only time will tell. For its part, CMD has made GEOS compatibility a staple of the new unit, so that OS will run correctly. At least one commercial venture, PROTOVISION, is allegedly planning an all GUI OS for the unit. See Newsfront (Section: news) for more information.

Brett Tabke, of PHD Software Systems, announced that he is busy upgrading his Karma assembler for the 128 to run on the new unit and praised the virtues of the new opcodes, modes, and options available when operating the unit in "Native mode." This started some discussion on writing code that only runs on the new processor. The sides were split almost immediately, as all noted that while the new applications would run very well on the SCPU, they would not run at all on a stock 64. Proponents noted that new software demands the purchase of the SCPU, while purists maintained that that would limit the software to a small market segment.

@(A): The "Virtual 1541"!

Now, before you start thinking of 3-dimensional plastic cases and track 0 head knocking in quadrophonic sound, let's explain the topic. Many users have expressed a desire to virtualize the IBM PC as a glorified 1541 drive with full emulation. The closest thing as yet is the 64NET package, which allows you to load and save programs to the IBM PC hard drive like it was a regular CBM drive. The drawback to 64NET is its non-standard access method. Before you can access the emulated CBM drive, you have to load special software on the 64 and use a special user port cable. So, for users who demand perfect emulation, no choices exist yet. The lack of options haven't dented the dreams of many who outlined the "technical specifications" of such a virtual disk drive.

```
================================================================================

@(#)toolbox: Menu Toolbox III
             by Jeff Jones (jeff@loadstar.com)
             Copyright 1996: J and F Publishing

@(A): Introduction

This toolbox has the most versatile menu commands I've ever coded, and
will make your BASIC and ML programs scream with speedy scrolling menus
and file selectors with multi- item selectability. That's right. I said
scrolling! Because of its size, there are only two versions of MENU
TOOLBOX III:

Filename           SYS Location  Notes

MENUTOOLS 1000      4096         (Reference: code, SubRef: menucode1)
MENUTOOLS 8000     32768         (Reference: code, SubRef: menucode2)

These are the optimal locations for BASIC. It's 8K (33 blocks) in length,
but your programs will be much smaller, and have access to RAM under ROM
for array, screen and menu storage. You won't suffer for memory.

There are four types of menus:

o Custom Item Menus
o File Requestors
o Screen Menus
o Instant Pages (from BASIC only)

@(A): ML and Compiler Users

You usually can't use SYS 32768,1,2,3,4,5 from ML or compiled programs so
you'll have to POKE the parameters into a location and then tell the
toolbox where to get the parameters. You tell MENU TOOLBOX II where to find
the parameters by loading the .X and .Y registers with the low and high
bytes of the location and then SYSing. From BASIC, the .X register is
location 781 and the .Y register is 782. Here I'll use $033C (+828) as an
example. The high/low bytes for location 828 are 3 high and 60 low. The
following BASIC example of the lattice command works equally well as BASIC
or compiled.

   poke828,x1
   poke829,x2
   poke830,y1
   poke831,y2
   poke832,t1
   poke833,t2
   poke834,c1
   poke835,c2
   poke781,60
   poke782,3
   sys32768+102

This may look slow, but in compiled programs, POKE commands are executed
at near ML speeds. If there is a chance that you're going to compile your
program, you may want to use the ML/compiler protocols from the start. It's
heck converting a program. I know (Directomeister).

ML programmers can embed parameters within their programs. It's actually
easier from ML than compiled:

   background ldx <b'lattice
              ldy >b'lattice
              jsr 32768+102
              rts
   b'lattice  .byt 0,39,0,24,9,10,15,12

@(A): Instant Pages

   SYS 32768+198,PAGENAME$, items,list$..., hotkeys$

Since it's by far the easiest to use, we'll discuss instant pages
first. A page is a computer screen with a user interface setup. Instant
page allows you to create with one SYS a screen with a tiled background, a
title bar, and a centered working menu complete with hotkeys. Consider the
following code:

   10 T$="M  Y   D A T A B A S E"
   20 A$(1)="OPEN DATABASE       (O)"
```

```
30 A$(2)="DEFINE FIELDS       (D)"
40 A$(3)="ADD RECORD          (A)"
50 A$(4)="SEARCH RECORDS      (S)
60 A$(5)="PRINT RECORDS       (P)"
70 A$(6)="RETURN TO LOADSTAR  (Q)"
80 A$(7)="odaspq"
90 sys32768+198,T$,6,A$(1),A$(2),A$(3),A$(4),A$(5),A$(6),a$(7)
100 onf%gosub200,300,400,500,600,700
110 goto 10
```

Make sure the number of items in your list$ matches the number declared with ITEMS. If you RUN this program, a typical LOADSTAR type program will pop up on the screen:

```
M Y   D A T A B A S E
OPEN DATABASE       (O)
DEFINE FIELDS       (D)
ADD RECORD          (A)
SEARCH RECORDS      (S)
PRINT RECORDS       (P)
RETURN TO LOADSTAR  (Q)

CRSR/RETURN TO SELECT
```

You'll have a CRSR/RETURN menu with hotkeys for each menu item, and more hotkeys if you simply include them. The menu is automatically centered left to right and top to bottom. CRSRing to ADD RECORD or pressing [A] will exit the menu and the variable, F%, will be 3. This is how you know which item or hotkey was selected by the user. There can be up to 40 hotkeys defined so that your page goes beyond the items presented on the menu. If hotkey #20 is pressed, the menu is exited, and F%'s value is 20. It's not mandatory for your menu to have alternative hotkeys, but if you do, list the menu's hotkeys first, and in the same order that they appear in the menu so that F% is always the same as the menu choices and the hotkeys.

Lines 10-80 set up the data for the menu into variables since you can see that there's no way all this text could fit on one line. Just imagine it with 20 hotkeys.

Line 90 calls the routine. Program flow is transferred to MENU TOOLBOX II until a menu item or hotkey is pressed.

Line 100 is one way of dispatching flow of the program based on the user input.

Ironically, this feature will only work from BASIC. I wrote the INSTANT PAGE feature as a simple driver to test MENU TOOLBOX II's ability to take commands from ML. I liked the routine I ended up with and decided it should be added to the package. Adding links for ML for this routine could be done -- but since I wrote the routine with many parameters intending for it to be called from BASIC, it becomes more difficult, even convoluted, to write ML links for a program which is essentially a series of ML links. Plus there's no time left this month.

```
@(A): Instant Page Setup: SYS 32768+201, tilea, tileb, colora, colorb,
                                title color, highlight color, menu color,
                                message color, frame color, frame on,
                                background, border
```

You can use my bland default colors or you can use your own. This command simply changes the presets.

Your background is made up of a mesh of two characters, A & B, in two colors. TILE A is one of the characters in the tiled background (0-255) TILE B is the other tile in the background

COLOR A is the color of TILE A, 0-15 where COLOR A has the following effect:

COLOR A VALUE	COLOR
0	BLACK
1	WHITE
2	RED
3	CYAN
4	PURPLE
5	GREEN
6	BLUE
7	YELLOW
8	ORANGE
9	BROWN

```
      10        LIGHT RED
      11        DARK GRAY
      12        MED GRAY
      13        LIGHT GREEN
      14        LIGHT BLUE
      15        LIGHT GRAY
```

This chart applies to all subsequent color values mentioned in the documentation.

COLOR B is the color of TILE B

TITLE COLOR is the color of the title of the page.

HIGHLIGHT COLOR is the color of the highlight bar of the menu

MENU COLOR is the color of the menu

MESSAGE COLOR is the color of the message at the bottom of the screen, "CRSR/RETURN To Select."

FRAME COLOR is the color of the frames surrounding the title and menu.

FRAME ON turns the frame around the menu on with a nonzero value. You might need to turn off the frame to squeeze in extra menu items. Maybe you don't like frames, either.

BACKGROUND is the background color

BORDER is the border color

@(A): Change Message: SYS 32768+204,New Message$

In case your page needs a custom message at the bottom of the screen, change it here. Keep it less than 38 characters.

@(A): Screen To Menu: SYS 32768+63,y,x1,x2,n,t,h,"keys"

Screen To Menu is an easy way to create a CRSR/RETURN menu from a vertical list of options that you've previously printed on the screen. So any list on your screen can be made to "spring to life" as a CRSR/RETURN menu. There may be up to 24 items, as wide as the entire screen. It's up to you to write the subroutines that correspond to each menu item.

After an item is selected, the variable, F%, tells you which item was selected. It will hold a value between 1 and the number of items in the menu. Here it is in use:

```
   150 sysaddr,y,x1,x2,n,t,h,"hotkeys"
   160 on f% goto200,300,400,500,600...
```

"f%" is the nth item selected.

Y is the starting row on the screen.

X1 is the left extreme of the highlight bar.

X2 is the right extreme of the highlight bar.

N is the number of items in the menu.

T is the text color of the unhighlighted items in the menu.

H is the highlight bar color.

NOTE: If you don't want the highlight bar to reverse items, add 128 to the color codes (0-15) of parameters T and H.

The MENU command has been extended per imperial order of Maurice Jones. Maurice likes to press one key instead of using CRSR/RETURN menus, which force you to press more than one key. The new "keys" field allows you to define hotkeys for each menu item, and also for flow control beyond the CRSR/RETURN menu.

For instance, you have five items on your menu, but "keys" is defined as "12345qlprt". 1-5 happen to be hotkeys for the menu in this example. You can use mnemonic keys if you like. If the user CRSRs to item 4 and presses RETURN, F% will become 4. If they press 4, f% becomes 4. But as a bonus, if the user presses "l", which is seventh in the KEYS string, F% becomes 7. If they press "r", F% becomes 9, and so on. MENU now has the power of BRANCHER. You can have up to 40 hotkeys in the string.

What would you use these hotkeys for? Well you might have a CRSR/RETURN menu, but also "q" to quit and SPACE to go to another menu. The only new requirement is that you now have to define hotkeys for each of your menus. It's a good idea to let your users know about the hotkeys.

To use screen menus from ML:

```
ldx <parameters
ldy >parameters
jsr 32768+162
```

User input is returned in 253

Parameters is a stream of legal parameter bytes followed by the length of the hotkey string, followed by the hotkey string.

@(A): Introduction to Scrolling Menus

These menus are more involved to implement, but well worth the setup time. The multi-file selector in DISKMEISTER is an example of a quirky BASIC/ML slow POKE (pardon the pun). You need self-contained ML to handle your menus, especially scrolling ones. MENU TOOLBOX will create its own pointers for the menu items you define. Normally these pointers will be right at the end of the ML. Each item in your menu will take up 4 bytes. The pointers can extend under ROM and IO with no problem. If you don't want the pointers to be at their default location, you can change the array location with the following SYS.

@(A): Change Location: SYS 32768+27,location

>From ML

```
ldx <location
ldy >location
jsr 32768+126
```

Since the default pointers will begin around $a000, you will want to change the location if you have data such as a screen stored there. Typically you'll want the pointers somewhere where they won't cost you anything, which is usually under some ROM.

You'll also want to use this command if you want to switch between pointers for different menus.

@(A): Declaring Menu Items

Before MENU TOOLBOX can create a menu for you, you must either BLOAD a directory, an EDSTAR PRG text file with a list of menu items in it, or declare items from DATA statements. I've included tools to make this process easy. Since directories are so structured, they don't need pointers so all you have to do is tell MENU TOOLBOX where to BLOAD the directory:

@(A): Bload Directory: SYS 32768+39,"$:*",device,location

Before you can show a file requestor, you must first have a directory in memory. You can BLOAD a directory anywhere in RAM, even under IO at $D000-$DFFF. There is NO NEED to use the CHANGE LOCATION command after BLOADing a directory. F% holds the number of files, but keep in mind that counting starts from 0, not 1.

>From ML:

```
ldx <filename
ldy >filename
lda namelength
jsr 32768+138
```

the directory filename, usually "$:*" must be followed immediately in memory by the device number byte and then the low/high bytes of the load address.

@(A): File Requestor: SYS 32768+45,x,y1,y2,r,c,h,s,m

This command will pop up a scrolling (if necessary) menu on your screen. Your users will be able to CRSR up and down or page with the + and - keys. You should print this somewhere on the screen so that your users will know. Lotta parameters? Here's what they mean:

X is the upper left hand corner of the requestor box. Since directories

are always going to be about 32 columns across, you would never have a number more than 6 here.

Y1 is the top row of the menu, 0- 20.

Y2 is the bottom row in the menu. Y2 determines how much of your screen will be taken up by the file requestor.

R stands for REVERSE mode. If you want all the items in the menu to be printed in reverse, put a 1 here. Highlighted items will appear unreversed.

C is the color of the menu and all unhighlighted menu items.

H is the color of the highlighted (current) item.

S is the color of selected files (if the menu is in multi-file select mode.

M is for MULTI MODE. It allows the user to select more than one file. Make this a 0 for a normal single file and 1 for multi-file selection ability. Files are selected with SPACE or RETURN. To exit the requestor you must press F1. Your program should inform the user of this if they are in the multi file mode.

When in single mode the file name is returned in W$, the position in the directory is in I% and the block size of the selected file is in B%.

>From ML:

```
ldx <location
ldy >location
jsr 32768+144
```

Have parameters as a stream of bytes in the order and extremes presented above. When in single mode, .X and .Y point to the filename and .A holds the file length. Compiler users, .A is location 780, .X is 781 and .Y is 782.

You can find F% at 253, I% at $14 and b% at $22

@(A): Index Items: SYS32768+33,item

This command works best with bloaded directories, but can be used on normal scrolling menu data. It returns the filename or menu item string of ITEM into w$. When f% <> 0, it means the item or file has been selected. So when you have a multi file menu, do a loop to check each file. If f% <> 0, act on that file.

[Big note:] if you're hurting for string array space or suffering from garbage collection blues, why not store/load lists into menu item slots in high memory? You can use INDEX to check your hidden pseudo arrays, and DEFINE MENU ITEM (below) to store/flag items. Such arrays won't be dynamic, but they can come in handy.

>From ML:

```
ldx <item
ldy >item
jsr 32768+132
```

.X and .Y point to the returned string and .A holds the string length. F% is in locations 253 and 254 always.

@(A): Menus from BASIC Strings

Menu items can also exist in BASIC, but preferably in DATA lines and not in dynamic strings since some dynamic strings can change location after a garbage collection.

MENU TOOLBOX has to know a few things about your menu items before it can make a menu out of them. It has to know each item's place in the menu. Is it item number one or three hundred? If you have many items, you can tell MENU TOOLBOX all about your menu in a FOR loop with the following command. This isn't difficult to do. All you have to do is:

@(A): Define Menu Item: SYS32768+30,item$,index,selected

Here you're telling the system the place of a certain string in your menu. If you're doing this from an array, you can use a FOR loop.

ITEM$ will always be a string variable, probably from an array or from DATA read into a string. If your menu items are in DATA statements, you don't need to DIMension an array to hold the strings. You can READ the data into a throwaway variable like a$ and then

```
sys32768+42,a$,index,selected
```

in a FOR loop. If you will want to print the menu item somewhere outside of the menu, you will probably want and need an array. In any case, string DATA in arrays doesn't take up any memory except for pointers.

INDEX is just the item's place in the menu.

SELECTED is whether or not the item should be considered selected when the menu pops up. You can use this even when not in multi-select mode to show that some items are not available or to highlight items to show that some pending action is needed.

>From ML:

```
ldx <parameters
ldy >parameters
lda string'length
jsr 32768+129
parameters .asc "item name"
.word index
.byt selected
```

@(A): Menu COmmand: SYSaddr+42,x1,x2,y1,y2,r,c,h,s,o,l,m

This command will pop up a scrolling (if necessary) menu on your screen. Your users will be able to CRSR up and down or page with the + and - keys. You should print this somewhere on the screen so that your users will know. Here's what the parameters mean:

X1 is the upper left hand corner of the requestor box.

X2 is the right extreme of the menu. Any menu item that exceeds the width defined by X1 and X2 will be truncated.

Y1 is the top row of the menu, 0- 20.

Y2 is the bottom row in the menu. Y2 determines how much of your screen will be taken up by the menu.

R stands for REVERSE mode. If you want all the items in the menu to be printed in reverse, put a 1 here. Highlighted items will appear unreversed.

C is the color of the menu and all unhighlighted menu items.

H is the color of the highlighted (current) item.

S is the color of selected menu items.

O is the offset, in case you've sectioned off your menu items, generally zero. If you want to pop into the menu at a particular point (say at the last item selected which you've preserved), use offset to do so. The rest of the menu is still accessible.

L is the limit or the highest menu item allowed. Note that this will usually be one less than f% when you use the Rack Em Up command.

M is for MULTI MODE, which allows the user to be able to select more than one item. Outside of a file requestor, there are few uses for this. Make this a 0 for a normal single item and 1 for multi-item selection ability. Items are selected with RETURN or SPACE. To exit a multi-select menu you must press F1. Your program should inform the user of this if they are in the multi item mode.

When RETURN is pressed in single mode, the item number is returned in F%.

>From ML:

```
            ldx <parameters
            ldy >parameters
            jsr 32768+141
 parameters .byt x1,x2,y1,y2,,r,c,h,s
            .word o,l
            .byt m
```

@(A): Set Mode: SYS32768+48,n

Sometimes you may have a regular menu and a directory set up in memory. You may want to set the proper mode for FIND AND CLEAR with this command before using it. N is 0 for normal retrieval and 1 for file requestor retrieval.

@(A): BLOADT Text File: SYS32768+15,file$,device,location

This command will BLOAD a standard EDSTAR text file into memory and place a zero at the end of the file to mark the end. It can also be used as a regular BLOAD.

FILE$ is the filename of the file you want to BLOAD.

DEVICE is any legal device number.

LOCATION is anywhere in RAM except the IO area $D000-$DFFF. Again, you really should take advantage of areas outside of BASIC and beneath ROM.

You can indeed use packed text if you BLOAD it with DTEXT, published on LOADSTAR #122 and the COMPLEAT PROGRAMMER.

>From ML:

```
              ldx <parameters
              ldy >parameters
              lda filename'length
              jsr 32768+114
   parameters .asc "filename"
              .byt device
              .word location
```

@(A): Rack 'em Up: SYS32768+36,location

This command simply itemizes every line in a text file that you've BLOADed so that it can be used as a menu. Pointers for each line in the text file will be located starting from the end of the file. So don't BLOAD a file that ends too close to $FFFF or the beginning of important data.

F% tells you the number of lines it has itemized. Save this into another variable because you will need the value to set a limit (bottom) to any menu made with the BLOADed data.

The longest length of a line is stored in $14 (decimal 20). If you want your resulting menu properly centered, check out this location immediately after the call as this location is heavily trafficked by Menu Toolbox and BASIC.

>From ML:

```
   ldx <location
   ldy >location
   jsr 32768+135
```

@(A): Text File Reader

You would simply BLOAD a text file using the aformentioned command, and then RACK IT UP. Next just define a regular menu wide enough to accommodate the text anywhere on the screen. Don't forget to prompt your users to press RETURN to exit the reader (which is really a big menu).

@(A): Report Location: SYS32768+54

If you don't want to rack up a text file again, you can find out and store exactly where the menu pointers begin with this command. The location will be reported in F%. Remember that F% is an integer variable, and has a max of 32767. So if F% is negative, subtract it from 32768:

```
   SYSAD+69:ad=f%:iff%<0thenad=32768- f  %
```

>From ML:

```
   jsr 32768+153
```

Location returned in .X and .Y

@(A): Get Word: SYS32768+51,color,cursor,limit,text$

Your program will probably need input from the user. Here you have the best input routine I've ever done. It allows you to cursor through already-typed text, and it allows you to predefine the text from a variable or static string, allowing the user to just hit RETURN if they don't want to change the text as it appears at the blinking cursor. The results of editing (or non-editing) is returned in W$.

COLOR is the color of text that is typed by the user.

CURSOR is the color of the flashing cursor.

LIMIT is the maximum length of the input.

TEXT$ will usually be null. If not, the contents of the variable will be printed at the current location of the cursor, respecting the COLOR parameter. The length of TEXT$ overrides LIMIT only if it's longer than LIMIT.

>From ML:

```
              ldx <parameters
              ldy >parameters
              jsr 32768+150
              rts
   parameters .byt color,cursor,limit
              .byt length of default string
              .asc "default string"
```

Returns location of edited string in .X and .Y. .A holds the length of the string.

@(A): Start Memory Print: SYS32768+60,location,string$

@(A): Continue Memory Print: SYS32768+63,string$

Use START MEMORY print to start a list of text in memory. This command allows you to print to memory in case you want to build a menu from software instead of a table or file. You can print a list in memory, RACK it up, and make a scrolling menu out of it. Works beneath ROMs, too. A carriage return is appended to the string in memory.

Once you've started memory printing, just keep sending strings with continue memory print. The updated address of the end of your menu is kept in locations 251 and 252. USE NO OTHER COMMANDS while building your list with memory print! You might corrupt 251 and 252.

The final item in your menu should end with a character string 0 so that RACK IT UP knows where the end of your list is.

```
   sys32768+63,final$+chr$(0).
```

START From ML:

```
   lda <string'loc
   sta $22
   lda >string'loc
   sta $23
   lda string'length
   ldx <start
   ldy >start
   jsr 32768+156
```

CONTINUE From ML:

```
   lda <string'loc
   sta $22
   lda >string'loc
   sta $23
   lda string'length
   jsr 32768+159
```

@(A): Scroll Up: SYS32768+18,x1,x2,y1,y1

This scrolls a section of the screen defined by the extremes above. Scroll need not be used with menus. That's done automatically, but since the code exists for the MENU command, I thought I'd give you direct access to it if you want. The cursor is placed right where new screen text would appear according to the parameters of the scroll.

>From ML:

```
              ldx <parameters
              ldy >parameters
              jsr 32768+117
              rts
   parameters .byt x1,x2,y1,y2
```

@(A): Scroll Down: SYS32768+21,x1,x2,y1,y1

This scrolls a section of the screen defined by the extremes above.

>From ML:

```
              ldx <parameters
              ldy >parameters
              jsr 32768+120
              rts
   parameters .byt x1,x2,y1,y2
```

@(A): Clear Row: SYS32768+24,code,color

In case you have to blank a duplicated line in the scroll area, this command will do it using the screen code and color you specify. Cursor position isn't changed.

>From ML:

```
   ldx code
   ldy color
   jsr 32768+123
```

@(A): Titles

In order to make use of the internal tiles, you MUST use a custom font in your program. The use of fonts in programs is beyond the scope of this article. FOr information on fonts, please see LOADSTAR's Compleat Programmer or Font Finale on LOADSTAR #92.

Here is the command structure of the tile functions. Since I made three versions of the program, I'll refer to only the $C000 version. Replace 32768 with the other addresses if you will be using other versions.

@(A): Lattice: SYS 32768,x1,x2,y1,y2,t1,t2,c1,c2

This command will create a colorful pattern on the screen. If T1 and T2 were 1 and 2, the grid would consist of As and Bs in the following fashion:

```
   abababababababa
   babababababbab
   abababbbabababab
```

These lattices can be any dimension, as thin as a column or row and as large as the entire screen. They can be very colorful and eye-catching as you will see in DIRECTOMEISTER on our pass-around issue, available from our web page at http://www.loadstar.com/

X1 is the leftmost column of the lattice, 0-39.

X2 is the rightmost column of the screen, 0-39 > X1.

Y1 is the top row of the lattice, 0-24.

Y2 is the bottom of the lattice, 0-24 > Y1. Note that if Y2 is greater than 24, the lattice can overwrite your BASIC program at $0801 (+2049).

T1 is the screen code of the tile in your lattice, 0-255. You can find the screen code of any character by printing it at HOME and then issuing the command:

```
   print peek(1024)
```

T2 is the screen code of the other tile in your lattice. It can be the same as T1 if you like. You can still get a lattice effect by using the same tile with a lattice of color only.

C1 is the color of T1, 0-15.

C2 is the color value of T2 in your lattice. As with tile selection, C1 and C2 can have identical values with T1 and T2 having different values, creating a lattice of tiles and not color. If T1, T2, C1 and C2 are all the

same, then what you have is a block. There's a simpler way to get a block.

>From ML:

```
          ldx <params
          ldy >params
          jsr 32768+102
          rts
   params .byt x1,x2,y1,y1,t1,t2,c1,c2
```

@(A): Block: SYS 32768+3,x1,x2,y1,y2,code,color

This routine will draw a block of characters on the screen. It will also paint an area of the screen without changing the characters if you use a CODE of 255. It can erase portions of the screen if you use spaces, and it can draw single lines or rows of characters if you like. It is a mainstay of my programming. Can't do without it.

X1 is the leftmost column of the block, 0-39.

X2 is the rightmost column of the screen, 0-39 > X1.

Y1 is the top row of the block, 0-24.

Y2 is the bottom of the block, 0-24 > Y1. Note that if Y2 is greater than 24, the block can overwrite your BASIC program at $0801 (+2049).

CODE is the screen code of the tile in your block, 0-254. You can find the screen code of any character by printing it at HOME and then issuing the command:

```
   print peek(1024)
```

When a CODE of 255 is issued, the BLOCK program will not alter the screen at all, only the color. This allows you to change (PAINT) colors of portions of the screen without having to re-print them. If you must use character #255 on the screen, use LATTICE with both screen codes set to 255. With both codes and both colors set the same, LATTICE becomes BLOCK.

COLOR is the color of CODE, 0-15.

>From ML:

```
   ldx <params
   ldy >params
   jsr 32768+105
   rts
   params .byt x1,x2,y1,y2,code,color
```

@(A): Outline Box: SYS 32768+6,x1,x2,y1,y2,color

This routine will pop up a box on the screen using the CMDR-A,S,Z,X, SHIFT-ASTERISK and SHIFT-MINUS characters. The area inside the box will not be affected.

X1 is the leftmost column of the box, 0-39.
X2 is the rightmost column of the screen, 0-39 > X1.

Y1 is the top row of the box, 0-24.

Y2 is the bottom of the box, 0-24 > Y1. Note that if Y2 is greater than 24, the box can overwrite your BASIC program at $0801 (+2049).

COLOR is the color of the box, 0-15.

>From ML:

```
          ldx <params
          ldy >params
          jsr 32768+108
          rts
   params .byt x1,x2,y1,y2,color
```

@(A): Copy Tile: SYS32768+9,FONT LOCATION,TILE,CHAR

This is a magic command. Within the ML are 60 tiles, lifted from TILE STYLIST. This command will copy any of those tiles to any character in your font. All you have to do is tell the command where your font is.

FONT LOCATION is the location of your font in memory.

TILE is the built in tile 0-9, that you want to have copied into your font.

CHAR is the screen code of the character you want replaced with the tile.

>From ML:

```
         ldx <parms
         ldy >parms
         jsr 32768+111
         rts
   parms .word font'location
         .byt tile,char
```

@(A): More Than Sixty Tiles?: SYS32768+12

If you use this command, you must be a tile fanatic. You can use values greater than 99 for TILE if you BLOAD a tile font into the proper location. This will only work with the $9000 and $0800 versions of TILE TOOLBOX. It can work with the $C000 version, but under normal circumstances, you can't BLOAD to the $D000 area, which is where the VIC chip and I/O are.

To get the proper BLOAD location, SYS32768+12. The address will be returned in the .X and .Y registers. For BASIC users, the address will be returned in locations 781 and 782.

ML users would simply insert the JSR before a LOAD:

```
   jsr 32768+12
   jsr LOAD
```

BASIC users would BLOAD their font into place this way:

```
   20 sys57812"tiles",8,0:poke780,0: sys32768+12:sys65493
```

This would give you a total of 266 tiles in memory if your tile font is 9 blocks long, but you will only be able to access the first 256, including the ten already in the ML.

@(A): Animation

You can use the COPY TILE command for animation by copying a series of tiles into the same CHAR. This will be much faster than drawing new characters or re-drawing an entire lattice or block. Of course if you're changing just one character, you can poke one byte on the screen, but the advantage of copying is that you have information [outside] of your font so your font isn't crowded and barely usable for text.

@(A): Shade: SYS 32768+96,x1,x2,y1,y2

SHADE paints with intelligence. I have always used BLOCK to shade an area with a uniform color just beneath and to the left of a box I'm about to draw. This gives a nice 3-D effect. SHADE does one better. If there are multiple colors in the area, each color is assigned a darker shade. So your overlapping windows will have a consistent shading effect.

SHADE can have a flash-type cycling effect when used repeatedly on the same section of a screen. Eventually it will only cycle between blacks and all the shades of white.

>From ML:

```
         ldx <parms
         ldy >parms
         jsr 32768+192
         rts
   parms .byt x1,x2,y1,y2
```

@(A): Screen Stash: SYS 32768+66,page

@(A): Screen Restore: SYS 32768+69,page

@(A): Screen Merge: SYS 32768+87,page

Without screen swapping, pop-up help screens and menus would be cumbersome. These two routines will store a screen at any page (256 byte section) in memory. To store the current screen at $d000 (53248), you would:

```
   SYS 32768+66,53248/256
```

To get the screen back:

```
SYS 32768+99,53248/256
```

$D000 is at page 208. Note that each screen takes up 8 pages, and should be kept 8 pages apart.

Note that screen stash stores the current border and background colors, as well as cursor position per imperial order of Maurice Jones. Screen restore will bring back the border and background colors of the stored screen.

SCREEN MERGE will restore a screen "under" an existing screen, meaning that every place where there's a SPACE can be written to by the screen being merged. Background and border colors aren't changed by MERGE.

In case you're confused about what screen merge does, MERGE would be a simple screen restore on a blank screen. On a screen with something, anything that's not a SPACE or SHIFT SPACE, MERGE allows restored screen to bleed through without wiping out the current screen.

You can use MERGE in games like Concentration, where you only want to reveal part of the screen at a time.

STASH From ML:

```
lda page
jsr32768+165
```

RESTORE From ML:

```
lda page
jsr32768+168
```

MERGE From ML:

```
lda page
jsr32768+183
```

@(A): Link: SYS 32768+72

Though all TOOLBOX routines that affect the screen use LINX and keep line links clear, you can call LINX directly with a SYS 32768+72. Same from ML.

@(A): Print At: SYS32768+75,x,y,string$

This will print text anywhere on the screen. X and Y are your screen parameters. STRING can be a literal or a string variable.

>From ML:

```
lda <string'loc
sta $22
lda >string'loc
sta $23
ldx row
ldy column
jsr 32768+171
```

>From ML the string must terminate in a zero.

@(A): Center: SYS 32768+78,row,string$

Centers a string (fewer than 41 bytes) on a specified line.

>From ML:

```
lda <string'loc
sta $22
lda >string'loc
sta $23
ldx row
jsr 32768+174
```

String must terminate in zero.

@(A): UPcase: SYS 32768+81,variable$

Converts all characters in a string to upper case. "We, The People" becomes "WE, THE PEOPLE"

Note UPCASE will not print the string for you.

>From ML:

```
   lda <string'loc
   sta $22
   lda >string'loc
   sta $23
   jsr 32768+177
```

String must terminate in zero.

@(A): LCase: SYS 32768+84,variable$

Converts all characters in a string to lowercase. "We, The People" becomes "we, the people". Great for use before a test of input like Y and y or N and n to consistently lowercase input.

>From ML:

```
   lda <string'loc
   sta $22
   lda >string'loc
   sta $23
   jsr 32768+180
```

String must terminate in zero.

@(A): CHARACTER SWAP: SYS32768+90,a,b,c

CHAR SWAP will search the screen for parameter A and change it to parameter B with the color, C. Here A and B are screen codes as revealed in LOADSTAR LETTER #34 or page 376 of your Programmer's Reference Guide. But the quickest way to find out a screen code is to print the character in the HOME position and then:

```
   print peek(1024)
```

or if your screen is moved:

```
   print peek(peek(648)*256)
```

In a flash, you can change every instance of a character to another character, or you can leave the character the same and just change its color. This is good for font animation where absolute speed isn't a factor.

Note: if you want to change characters, but not their colors, send a color code of 128.

>From ML:

```
         ldx <parms
         ldy >parms
         jsr 32768+186
   parms .byt a,b,c
```

@(A): COLOR SWAP:SYS32768+93,c1,c2

This changes every instance of a target color on the screen to another color.

>From ML:

```
         ldx <parms
         ldy >parms
         jsr 32768+189
   parms .byt a,b,c
```

@(A): BRANCHER:SYS32768+99,"string"

BRANCHER adds very quick flow control to BASIC programs. Send this routine a string and it will wait until a key, included in that string, is pressed. The string can be up to 207 bytes long. Imagine the BASIC code necessary to check 207 hot keys! This call will handle it in one command -- and much, MUCH faster.

When a valid key is pressed, BRANCHER will inform you which key was pressed by passing its instring position to the F% variable

```
   1000 SYS 32768+99,"mdvc":on f% goto100,200,300,400
```

Here's the BASIC equivalent:

```
   1000 geta$:if a$<>"m"anda$<>"d" anda$<>"v"anda$ <>"c"then1000
   1010 ifa$="m"then100
   1020 ifa$="d"then200
   1030 ifa$="v"then300
   1040 ifa$="c"then400
```

Not only is this code more bulky, but imagine how clunky it would be if you had 25 active keys.

>From ML:

```
   ldx <string'loc
   ldy >string'loc
   stx $22
   sty $23
   jsr 32768+195
```

String must terminate in a zero.

@(A): License Agreement

As with all LOADSTAR tools, we encourage you to use them in your programming endeavors. You can do so without a licensing fee as long as you mention somewhere briefly in the docs that you got the routines from LOADSTAR. If you'd prefer not to mention us, write for permission. LOADSTAR's tools may be used in commercial products, and programs published in other magazines under the same provisions.

==========

@(#)fido: FIDO's Nuggets
by Geoff Sullivan (sunfish@gis.net)

Summer slows things down everywhere, except perhaps in Zone 3, and the Commodore FIDO Echos are no different. I like to think we Commodore users have other interests beside our computers, and that we pursue them at this time of year. Message flow was slow, and at times sporadic. There was some discussion of a hitch in the system, and even now, I've been over a week without one new message arriving at my node. There is also concern that the Internet has absorbed some of the Echo traffic too. More of us have access to it, and there is no doubt that it's faster.

@(A): Warning: Killer App. Movie at 11.

The C128 killer application with a life of it's own, QWKRR, by Rod Gasson, is in version 5.0b at this writing. There are known bugs in it and Rod has posted the symptoms and cautions frequently on the Echos. Users are able to respond and help Rod work out the bugs from his Australian lab, while most of us in the Northern Hemisphere are enjoying a warm summer!

@(A): The Canon Is of Little Use, Sir.

Traffic has been light in the Geos Echo, but a major topic has been the use of Canon InkJet printers with Geos. Manufacturers are now catering more and more to Microsoft's Plug-n-Play concept, while we plug-n-pray. Only the older InkJet printers, such as the BJ-200 and 600 series, have actual DIP switches for manual configuration. Until software is available for us to configure the newer printers, their use will be limited.

@(A): Catch What WAVE?

Discussion of Maurice Randall's Wave for Geos 128 has been centered around speculation as to what it will end up being. Will it be a SLIP/PPP connection to the Web, or a high speed terminal program? Only Maurice knows for sure, and lately he's been busy tweaking software for CMD's Super CPU, the C64 version of which has begun shipping.

@(A): Speed Thrills....

Speaking of the Super CPU, the few users that have them by now are raving. I expect the message base to have more information as more SCPU's are put into service. We have all read the hype before, but actual user experiences posted on the Echos are confirming the awesome speed of this accelerator cartridge.

@(A): Ingenious Commodore Usage

Finally, an often asked question is, "What do you DO with a Commodore computer?". There is much to do, and recently a thread addressing that subject was begun. John Davis' story of how he and his VIC-20 increased his

Union participation by using a mail-merge program to personalize meeting announcements brought a smile. He also has used his SX-64 to do sing-alongs for kids at a local campground. SID is his only instrument!

So, that's a glimpse into the world of FIDO, the wonder dog of networks, for this time.

Here, boy....

==========

@(#)review: The Hacking Review
by Tom Warnes

@(A): The Compleat Crossword, published by J and F Publishing.

This is a collection of crossword games which are made a little simpler because the computer is watching to make sure you don't cheat. To start, simply load the disk just like any LOADSTAR offering: LOAD"!",8,1 (substitute 8 with whatever device you are accessing). You'll find most of the actions and key mappings identical to those found on the LOADSTAR menu program.

The start up screen gives you the option of finding out about the people are who created the puzzles in "All About This Disk". The other choices are "Change Title", "Change Music" and "Exit to Basic". The Change Title only allows you to change the background screen. The Change Music becomes a heaven sent option after you've been playing this game for a few hours, or even minutes. You may find yourself really hating the background music; therefore, I'll give a few tips on turning the music off completely. The music doesn't come on until you've loaded a puzzle into your computer off of the disk. To turn the music off completely hold down the CNTR key while pressing the S key or press the CLR/HOME key. The CLR/HOME key acts as a toggle as does the CNTR-S combination.

There were a few minor bugs on the disk that I found annoying. In the option "All About This Disk" only a summary of the creators of the puzzles is given. They don't attempt to explain the different functions or how to access them. My suggestion would be to add an extra subgroup called something like "Credit to the Creators" to put their names in, and in the "All About This Disk" put a more complete description of the choices available to the user. The "Change Title" had me stumped for a short while until I discovered it only changes the texture of the background screen, not the color to help someone who has been staring at the screen trying to solve these puzzles.

The puzzles themselves are interesting even if sometimes they don't stick to the theme of the puzzle. There is a small hiccup in the music presenter, as it had a small problem playing the last song on the list available to the user. It acted like a cassette player playing a cassette that has the tape wound to tight on the spools.

Except for these minor problems it was interesting having the computer keeping me honest. One big thing that has bothered me for years is the fact no one has found a way to reward the player for successfully completing a game on the computer. I've seen a simple message flashed on the screen or even a colorful display. This is not for everyone. A possible reward would be to have a timed series of puzzles so the player could compete against time. A feature I liked was the fact that you could mark a puzzle as having been completed, or if you're like me, one that you've worked on and still have to complete. You are given this option each time you go from a puzzle to the start-up screen. Most of these puzzles seem to have come from the mind of Barbara Schulak, a name you will recognize from the Loadstar 128 collection.

@(A): The Compleat Lee O., distributed by J and F Publishing.

Speaking of the Loadstar 128 collection the next group of programs came from those disks. It is called the complete Lee O. This is a collection of programs submitted to Loadstar to be put out on their 128 Loadstar disks. The programmer, Leo O. Clinton, tried to help fill the barren field of 80 column programming. They are all utility programs. Their titles are "Mutual Funds", "Resumes", "Genealogy", "Auto Expense Tracking", "Kitchen Upkeep" (Recipes), and "Home Finances". All of these programs come on one disk and it is highly recommended that you copy each onto a dedicated disk. To explain the working of the programs, I sometimes borrowed heavily from the text that Leo O. Clinton typed in to explain the workings of his programs.

@(A)subapp: Mutual Funds

Works with drive 8 only. Keeps 253 transactions in each of 16 Mutual Fund Accounts. Computes profit/loss, investment and total returns and investment and fund performers. Output can be to screen, disk or printer if you have included all the recommended files on your disk. Escape will take you one level back from the level you're working on. You can add accounts, delete accounts, modify accounts, or add transactions. This program works on the trade date of your accounts and must be entered in the request for date.

@(A)subapp: Resume Writer

This program is important in today's shifting job market. It works with a wide range of disk drives. It can create a resume after you type in the answers to a range of questions. It creates a resume to your personal preference. The chronological option lists last job first, or you can go with most important job first and on down to the least important. All information can be saved to disk and then recalled and printed when you need it.

@(A)subapp: Pedigree 128

Keeps track of 5 generations of your family. Gives you the option of creating a full page of information for each member. If you get stuck and don't know how to answer a question, the help screens will steer you in the right direction. After the title screen is displayed a chart will show you the relationship of the people you are entering. Drives 8 through 11 are supported. This is really a fine program for someone interested in his or her family relations.

@(A)subapp: Auto Expense

If you're one of the people you see at the gas pumps jotting down mileage, cost and quantity of gas in a little book, this program is for you. This program keeps a record of 18 cars with the possibility of 252 entries for each car per disk. The program is menu driven. After you enter data you can use what you have entered to create graphs or print a record for the car you pick. The graphs will show you Actual mileage, Cumulative mileage, or a Cents per mile Pie Chart.

@(A)subapp: Cook's Helper

This program is a electronic recipe box with a lot of extra features thrown in. It allows you to make up a weekly menu and will help make up a shopping list so you can get all the ingredients you will need. There is a conversion calculator to help you go from metric to American Standard, and also an option to convert to different quantities. Entering the recipes into this program is a little different, as you must use various buttons to access certain menus. For example, you press the F1 key whenever you want to print quantities. A list will appear and you use the cursor keys to select the proper quantity. I have used this program since it first came out on Loadstar 128 and it saves on the number of piles of paper I have laying around.

@(A)subapp: Fiduciary

The closest program that I can relate to this program is Sylvia Porter's Personal Finance 128 series put out by Timeworks. Sylvia Porter's has a few more small functions this program doesn't have. This program allows you to know what your transactions have been, balance your checkbook, create a budget and show you how close you are to staying within that budget. It shows you what Assets/Liabilities you have, and allows you to see in a chart where your money is going. A 30+ page manual that explains what functions this program is capable of is available for you to print out.

==========

@(#)hw: The CMD Nirvana: The Guts and Glory
by Todd Elliott (telliott@ubmail.ubalt.edu)
http://ubmail.ubalt.edu/~telliott/commodore.html

@(A): Introduction

Before you begin, download the CMD Nirvana picture off of my WWW site! Take a good look at the picture, and if it doesn't whet your appetite, this article will! This is the computer that I use at home for my various programming projects and good old fashioned entertainment. It consists of a C128DCR with three built-in foreign components: a 12 VDC fan, CMD HD 85 megabyte drive and a FD 4000 3.5 disk drive. The case is painted obsidian black. It does have a 4MB RAMLink hooked up with a 1750 REU and a Swiftlink or Action Replay cartridge. Other peripherals circling the CMD

Nirvana's universe are an 1541, 1571, a SlikStik, MW350 printer interface, in addition to a host of books and magazines dedicated solely to the c64/128 line. (Soon, it will have a SuperCPU 20!) It is a very powerful computer, but it doesn't cure my malady of starting programming projects, but only to leave them unfinished to do the next one. All of this power is corrupting, indeed!

@(A): Disclaimer:

This article, while resembling a how-to-do-it-yourself article, is only meant for entertainment and informational purposes and is to be used at the reader's own risk. Mr Elliott, Commodore Hacking, nor Brain Innovations, Inc. can be held responsible for any damages and/or injuries occurring as a result of following this information contained in this document. Electronic circuitry is fragile and can be damaged easily. Users must use ground strips or some other means of protecting their circuitry from accidental discharges of static electricity. Also, soldering equipment, electronic outlets, etc. pose a significant danger to self, and extreme care should be used in operating these items or working on electronic circuitry. Be sure to unplug the electronic circuitry before commencing work. I also disclaim any and all warranties, both express or implied.

Tremendous thanks goes to Al Anger, who, with his substantial assistance, made this CMD Nirvana computer a reality. Thanks, Al Anger! For those who don't know him, take a peek at Commodore World's Issue Ten cover, and you'll see his C128T (Tower) computer, plus a host of other hardware projects. But, Mr. Anger did not write this article and is not responsible for its contents, in which the disclaimer above still applies. You can reach Al Anger at coyote@bridge.net. Thanks also go to Max Cottrell, who did the scans of the photographs, courtesy of MC Photography, mcphoto@izzy.net. The standard disclaimer also applies to Max Cottrell and MC Photography.

@(A): A Verbal Tour

Let's begin with a visual inspection of the front panel: The plastic panel contains a sticker overlay for the CMD HD, which is located on the left side of the panel. The HD sticker overlay is located next to the C128D's POWER-ON LED on its right. The FD 4000 occupies the right side of the plastic panel. The right side of the plastic front panel is used to house a receptacle for the C128D's built-in 1571 disk drive, with a swinging gate. But, the built-in 1571 drive was removed, and the left side panel was cut and filed smoothly with the correct dimensions to fit a FD 4000 disk drive snugly. (Well,almost.)

Continuing with the visual inspection, we turn to the left side of the C128D. The metal casing was cut to allow for the fan, power cord and off/on switch to be accessible. If you note the picture carefully, the cuts on the case aren't in perfect sync with the interior. I guess the design won't be winning any art awards anytime soon. ;) The main thing is that I can access the off/on switch, and be able to plug/unplug the power cord without any difficulty. As for the fan opening, a circular opening as shown in the picture is not needed. In fact, a wave of lines cut on the case would be sufficient to allow air circulation. Going to the back end of the C128D, you will see wires snaking out of the receptacle in which the power cord formerly went through.

@(A): Opening the Hood

Let's dispense with the visual inspection and actually open the 'hood' of the CMD Nirvana computer! Yes, I know it's not a pretty sight and may look daunting to you, but it can be done! From the top view, it looks like the power supply for the C128D has been moved from its original position 90 degrees counterclockwise. The CMD HD motherboard now occupies the lower left corner of the C128D case. The FD 4000 juts across the lower right corner of the C128D front panel and the C128D case. As you can see, the internal 1571 disk drive has been ripped out. Wires are everywhere, but most of them settle down in the blank area in the upper right corner of the C128D case.

@(A): Digging In

Ah, where will we start? Let's take a closer look at the plastic front panel of the C128D. The CMD HD front panel was inserted and attached to the plastic front panel of the C128D, and is not attached to the metal case of the C128D. The marriage of the CMD HD front panel and the C128D's plastic front panel was accomplished by four screws and three pieces of plastic. First, I used masking tape to cover the CMD HD's metal case on the front side. This front side already has holes drilled on it by CMD or some OEM manufacturer. I used an X-Acto® knife to cut holes in the

masking tape, popping through the holes in the metal case. When finished with the poking of the holes in the masking tape, I have a 'drill image', so I peel off the tape from the CMD HD's metal case, and affix it to the C128D's plastic front panel on the left side. Making sure that everything's aligned, I proceeded to drill holes, following the 'drill image', in the plastic front panel. After peeling off the masking tape, and then doing a test fit with the CMD HD's front panel with its protruding LEDs, and swap buttons, it was a perfect fit! I then made the connection more solid with four screws and pieces of plastic, by attaching the CMD HD's front panel to the left side of the C128D's plastic front panel. Then, I superimposed the sticker overlay for the CMD HD perfectly with the LED's and the push-buttons, and used spray-on glue to make the overlay a permanent part of the C128D's plastic front panel.

@(A): Preparing the FD-4000

As for the FD 4000, I used a saw to cut off portions of the plastic front panel of the C128D where the internal 1571 disk enclosure used to be, cut in correct dimensions to fit the FD 4000. It was still rough, so I used a file to smooth the edges, on a gradual basis, until the FD 4000 made a snug fit. I measured the front cavity of the FD 4000 and used those same measurements to make the cut on the C128D's plastic front panel on the right side. As you can see from the picture, there is little room for error, especially at the top of the plastic front panel. Generally, the cut was made as far as to the right side of the C128D's plastic front panel as possible.

@(A): Preparing for the CMD Hard Drive

Now, to the internals of the C128D. I had to move the power supply counterclockwise 90 degrees to make room for the CMD HD. I quickly ran into a snag, as the power connector between the PCB and the power supply was too short. I had to cut off all the wires on the power connector, then enlarged it by soldering all wiring to a cannibalized wiring from a power supply wire to both ends of the power connectors. For further clarification, a close-up of both ends is supplied. The cannibalized power wiring was very convenient, as they were color marked, and it was easy for me to make sure that the wiring corresponded with each end of the power supply connectors. Next, I spliced the +-12 VDC wires on the power supply connector to supply continuous power to the fan. Usually a fan comes with black and red wires. Splice the black wire of the fan to the black wire of the power supply. Splice the red wire of the fan to the yellow wire of the power supply. Next, the fan was attached to the power supply with screws, to the right of the power cord connector.

The structural improvements to the PCB was made by bending the PCB power connector end at a 45 degree angle, for it would collide with the position of the CMD HD's PCB. Last, I connected one end of the entire power supply to the back edge of the C128D's metal case. This was accomplished by bending the end of the power supply at a 90 degree angle and screwing it together to the metal case of the C128D. However, one end of the power supply remained unsupported and would tip over onto the PCB, causing possible malfunctions for the C128D. So, a makeshift support was propped upon the C128D's PCB to support the remaining end of the power supply. This support, like all other supports used, were fitted with black electrical tape at the bottom to prevent shorts on the PCB. The massive heat sink covering the VIC - II chip and the 80 column display chip supported the other end of the power supply.

@(A): Installing the HD

With the power supply situated 90 degrees counterclockwise, we now have room to insert the PCB for the CMD HD 85 megger. I took the CMD HD's PCB out of its original metallic case that CMD supplied. (NOTE: If you do that, you may void CMD's warranty on the unit.) But the CMD HD's PCB could not just sit atop the C128D's motherboard, for there may be accidental electrical shorts or other problems. So, the CMD HD's PCB just hovered above the C128D's motherboard by approximately 1/4 of an inch. This was accomplished by erecting two support beams above the C128D's motherboard. The beams were obtained at a hardware store. They are aluminum and are easily malleable with pliers. I screwed down one end of the beam to the middle edge of the motherboard, leaving the other end of the beam covered with electrical tape. I could have screwed down the other end, but might have cut the motherboard in a way that Commodore never intended and screwed up the whole thing. All supports listed in this article are screwed down on existing 'drilled holes' in the c128d's PCB. Please note that the support beam is nearly aligned with the motherboard's power connector. I put electrical tape on top of the support beam, as this top will support the CMD HD's PCB. This is to ensure a smooth operation of the unit without any electrical shorts or other problems. Also, look at the rear view of the support beam in the

picture, for it can give you a clear idea of how tall it is in relation to the C128D's motherboard.

As for the other support beam, it is screwed down on both ends to the C128D's motherboard on the far left side. This support beam is taller than the one explained earlier. The reason is that the CMD HD's PCB will be screwed on that support beam, so this support beam would have to be aligned parallel to the top of the first support beam. Then the second support beam is covered with electrical tape and two holes are drilled at both ends, which align perfectly with the CMD's HD PCB's drilled holes. (See photo for further clarification.)

Finally, to attach the CMD HD's PCB to the two support beams, I made sure that both beams were in alignment and supported the PCB in a level fashion, almost parallel to the C128D's motherboard. I attached one end of the CMD HD's PCB, which has no cable connectors, to the second support beam on the far left of the C128D's motherboard. Two screws were used to make a secure connection to the support beam. The other end of the CMD HD's PCB, the one with all the cable connectors, rested on the first support beam, but was not screwed down. It could be done, but as the PCB was already secured, I didn't bother. The cables running out of the CMD HD's PCB are in a tight space, due to the FD 4000. As you may note from the picture, there is no room for the internal 1571. If you decide to do an internal CMD HD operation on your C128D, you must sacrifice your internal 1571 drive.

@(A): Installing the FD

With the CMD HD PCB out of the way, I began work on the FD 4000. Before the FD 4000 was inserted, I cut off pin one of the C128D's chip U113. This pin gives a signal that the internal 1571 drive is in existence. With that pin cut, the C128D no longer recognizes the existence of the internal 1571 drive. It will, however, recognize any peripherals using the serial bus, as I use a external 1571 as Drive #8 in my system. I took the FD 4000 mechanism and the motherboard (it's built as a single unit) out of its original case that CMD supplied. (NOTE: Doing so may void CMD's warranty on the FD x000 unit.) I used trial and error to position the FD 4000 drive in the C128D's metal casing. This is to ensure that the FD 4000 would not protrude too far out of the C128D's plastic front panel, or intrude too far inside the C128D's plastic front panel. As you can see from the following picture, the FD 4000 unit protrudes from the C128D's metal case by approximately 1 1/2 inches.

Lastly, I attached a support beam to the bottom right in front of the C128D's metal case. This involved some cutting on the edge of the bottom of the C128D's metal case in order to make the FD 4000 fit the C128D's metal case. The bottom edge of the C128D's metal case was not cut out completely, rather, it was bent inwards with pliers, creating a small surface for the FD 4000 to rest upon. Again, trial and error was used to determine how much to cut and bend the bottom edge, so that the FD 4000 unit would fit the C128D's metal case and the plastic front panel. Finally, the FD 4000 unit was screwed upon the support beam. This FD 4000 also rests upon the bent in bottom edge. This creates a stable surface for the FD 4000 unit to operate without having to insert a second support beam onto the C128D's motherboard to support the other edge of the FD 4000 unit.

Finally, I connected the cables to the CMD HD's PCB and the FD 4000, and attached the ribbon cables to the CMD HD's PCB, including that of the hard disk drive and the front panel controls. I extended the length of the power-on LED wire for the C128D's power supply. This extension was made by adding more wire and resoldering the connections. This was necessitated because the 90 degree turn of the power supply made all the wiring too short. :(Ah, the wisdom of CBM to cut costs by making everything short and sweet! I made sure that the CMD HD and the FD 4000's on/off switches were already in the ON position. Last, I bought Velcro® from an art supply store, and liberally applied it to the CMD HD 85 MB drive mechanism. The opposite end of the Velcro® attachment was pasted on the inside of the top metallic case of the C128D. This was positioned roughly 3 inches from the front edge of the top metallic case. Trial and error was used here to determine the optimum measurements. The CMD HD 85 mechanism was then attached to the inside of the top metallic case of the C128D unit by Velcro®. This is a safe method, for the CMD HD has been running for almost a year without any glitches. Close the case carefully (The HD is on the case, after all!), and attach all the power to a power control center with individual switches. The individual switches then can be used to control a specific peripheral such as the internal FD 4000 or the CMD HD 85, without having to open the unit to access the off/on switches, or drilling holes in the case of the C128D to attach these same switches. Convenient, isn’t it?

@(A): Lessons Learned

Before concluding the article, it seems possible that one could put a 1581 drive in lieu of the FD x000 drives. It's possible, but I do not own a 1581, so I really cannot comment. I would guess that the general guidelines for the FD x000 drives would apply to the 1581. Granted, all of this looks quick and dirty. But when I first started out with this project with Al Anger, we didn't have step-by-step manuals, or other references. It was truly a new territory for me. (Al has already done such hacks for his C= computers before me, and his experience was invaluable.) However, there were some bloopers. One, the c128d died upon powering up. It turned out that the fan +- 12 VDC wiring was connected to the wrong wire, and shut down the system. With that fixed, the c128d powered up okay, but now, the CMD HD died. If that wasn't frustrating enough, it was difficult to find out what went wrong, and I was wondering if the FD 4000 would be next. The culprit was a break on the CMD HD PCB. I accidentally drilled a cut on the PCB. With some soldering, I fixed it by building a bridge between the break in the CMD HD PCB. Whew! Everything now worked, and has worked since. Consider those bloopers invaluable lessons learned in dealing with sensitive electronic circuitry.

@(A): Conclusion

These are just general guidelines and are meant for entertainment purposes only. It serves as an inspiration to those who always dreamed of building their own C= beasts. I'm sure that there are countless users that have customized their computers, and with these general guidelines, undoubtedly, some users will want to create such a monster with rich C= 8 bit computing power!

@(A): Postscript

For those Commodore users who do not have access to graphical browsers, you can contact the author for a Word v7.0 document printout with pictures. The cost is $3 dollars for people living in the U.S. Canadian readers will have to pay slightly more in U.S. dollars.

==========

@(#)surf: Hack Surfing

For those who can access that great expanse of area called the World Wide Web, here are some new places to visit that are of interest to the Commodore community. In early 1994, when the US Commodore WWW Site started, the number of sites online that catered to Commodore numbered in the 10's. Now, the number is in the 100's. What a change.

If you know of a site that is not listed here, please feel free to send it to the magazine. The following links have been gleaned from those recently changed or added to _CaBooM! - Your One Stop Commodore Links Site_. (http://www.msen.com/~brain/cbmlinks/).

To encourage these sites to strive to continually enhance their creations, and because we like to gripe :-), we'll point out improvements that could be made at each site.

@(A): Companies

o Centsible Software
  URL: http://home.sprynet.com/sprynet/cents/
  Centsible Software has been a common name in Used Software Distribution for a while now. their WWW Site contains catalogs for the different platforms they support. C=Hacking gripe: Although the site is very informative and works well for either text or graphical browsers, it is rather plain. A bit of text here and some judicious use of HTML 1.0 tags would spice it up immensely. Text Browser compatibility shouldn't force WWW sites to be dull.

o Arkanix Labs, Inc.
  URL: http://www.arkanixlabs.com/
  Arkanix Labs recently purchased Threshold Productions International in order to expand their presence in the Commodore Market. They have a stocked WWW Site, complete with a catalog and recent news press releases. C=H gripe: For the graphical set, the site is heavy on graphics, although it does offer text link alternatives.

o Herne Data Systems, Inc.
  URL: http://ourworld.compuserve.com/homepages/herne_data/cpm.htm
  In the 1980's, HDS created some utilities for the C128's CP/M mode. Jugg'ler 128 allowed the CP/M user to read over 140 different

disk formats, while Scramb'ler 128 encrypted files and disks. Although HDS doesn't support these products anymore, they do provide them and numerous reference works on their WWW Site. C=H gripe: text based browsers might not handle the HTML TABLE very well.

@(A): Demo Groups

o Demolition
URL: http://www.cei.net/~rreed/
As the Webmaster puts it, " Demolition is currently more of a concept than anything." Demolition is slated to become a disk based demo programming resource magazine. However, at present, the site contains a few articles on topics like boolean logic and VIC internals. In addition, the site contains a comprehensive bibliography of basic and demo related programming articles. C=H gripe: The bibliography is on the main page, making it a lengthy piece of text.

@(A): Reference Works

o The Commodore Web Ring
URL: http://www.ncf.carleton.ca/~ag090/HomePage.ringpage.html
Although not a WWW site per se, this site starts you off on a journey through various Commodore WWW sites. It's entertaining and will undoubtedly take you somewhere you wouldn't have been before. C=H gripe: Graphical surfers might find the large "buttons" annoying.

o Info and Files for Commodore GEOS
URL: http://www.radiks.net/irv_cobb/geos2.html
This, a sub page under Irv Cobb's home page, offers various GEOS utilities that were either programmed by Irv or that he finds useful. GEOS users should bookmark this page. C=H gripe: As above. A bit of HTML 1.0 would spice this page right up.

o The History of Commodore
URL: http://www.geocities.com/SiliconValley/Heights/8329/History.html
A sub page of "The Commodore Haven", this sites gives the low-down on Commodore, from its inception to its demise. C=H gripe: well done page.

o Commodore Pictures
URL: http://www.swt.edu/~ez13942/commie/cbmpics.htm
Although the author of the page is not mentioned, he takes us on a tour of his various Commodore computer systems. It's always interesting to see how folks use their machine. C=H gripe: The graphics and the captions are all on one page, which makes the page a bit lengthy.

o The KIM-1 page
URL: http://www.magic.ca/~yhpun/ianpun.html
For anyone who doesn't know what a KIM is or how it relates to Commodore, this page is for you. Featuring pictures and explanations of both the KIM-1 and the author's home-made expansion board, this site offers a glimpse of the life of the 6502 pre-CBM. C=H gripe: The KIM-1 stuff is on the same page as the author's personal and business information.

o CBMSearch - a search engine for searching Commodore related software.
URL: http://www.ts.umu.se/~yak/cccc/cbmsearch/
Bookmark this site. Now. Offering a concise way to find Commodore software quickly and easily, CBMSearch can relieve the headache of trying to find CBM titles on the Internet. C=H gripe: Can't really say there is a gripe. The page is concise and could be spruced up a bit, but in this case, function overrules form.

@(A): User Groups

o Bronx User's Group (BUG)
URL: http://www.mediaworks.com/bug/
As Commodore Authorized User Group #0065, BUG's WWW site contains links to mail each of the officers and some links to other sites. A phone number is given for information. C=H gripe: The site's a bit low on content. We applaud the presence, but a map or some info about the group would be nice.

o Stone Mountain User's Group - Contains information on when and where we meet.
URL: http://www.cris.com/~Derektp/
Although supporting all orphaned computer systems, SMUG (love the acronym) appears to focus on the CBM systems and contains information on meeting places and times. C=H gripe: As above, the content is a bit low, but at least directions and times are given.

o ICPUG Independent Commodore Products User Group
URL: http://www.icpug.org.uk/

Although not devoted exclusively to the CBM 8-bit line, ICPUG does support them. A wealth of information, including the ability to join and club news is available. C=H gripe: For the graphical set, the site is a bit heavy on graphics. (For these sites, viewing with Lynx offers faster response.)

@(A): Individual Commodore Users

o Commie web page -- Better red than IBM
URL: http://www.swt.edu/~ez13942/commie/
Bo Zimmerman titles his page this peculiar way. He offers some history on his use of Commodore systems and offers some pictures of his various equipment. He also presents the Commodore Pictures page, detailed above. C=H gripe: the background makes for hard reading on some graphical browsers.

o Don's Page
URL: http://people.delphi.com/novan/
In an effort to offer people with text browsers some picture content, Don has included ASCII art in his page. We are impressed. Basically, the site details Don's hobbies and how the Commodore fits in. C=H gripe: We really think the ASCII is great, but there's no reason to make all the text in the document mono-spaced.

o Dave's Commodore 64 page
URL: http://www.csun.edu/~hbbuse08/c64.html
This page is for those looking for game information, SID tunes, and emulators. (Note: some of the files on this site are copyrighted.) C=H gripe: The top says "This page looks best when viewed with Microsoft's Internet Explorer or Netscape!"

o John Elliott's Home Page
URL: http://www.nsis.com/~prof/
John has a very extensive site detailing computers in education, including a piece on using "obsolete computers" in the classroom. A number of pieces available on the site include education conference highlights, and a discussion of how he finally hooked up to the Internet. The site is best viewed with Lynx, as it is enhanced for Lynx! (that's a switch) C=H gripe: The layout of the pages seems a bit haphazard, but maybe it's just us.

o Irv Cobb's home page
URL: http://www.radiks.net/irv_cobb/
If you want to know about Irv or where he grew up, it's all here. Hyperlinks are spread throughout the text to direct you to different topics. C=H gripe: none. Although it isn't as "splashy" as some pages, it is laid out well, and looks fine, although simplistic, on a graphical browser.

o Commodore 64 Online
URL: http://www.geocities.com/SiliconValley/Park/4645/
Tim Plelp's site is brimming with links to his favorite utilities and applications. Also included is a comprehensive list of PD/Shareware titles. C=H gripe: use unordered lists instead of "*" for the various items.

o Commodore Connection
URL: http://www.gis.net/~sunfish/crcbm.html
Geoff Sullivan's personal site offers articles and software to make using a C64 or 128 more enjoyable. Besides links to other sites, he also offers up Perfect Print fonts and articles on how to expand your REU to 2MB and how to build a simple RS-232 converter. Of special interest is a collection of customized GEOS mouse pointer icons and a program called QWIKSTASH that will copy files in GEOS on bootup. C=H gripe: Great resource, but little information about Geoff and how he uses his CBM.

==========

@(#)jb: Jim Butterfield: The Commodore Guru - An Interview
by Jim Lawless (jimbos@radiks.net)

@(A): Introduction

My initial interest in the Commodore 64 computer began in 1983. At the time, my primary source of information pertaining to the C64 came from Compute! and Compute!'s Gazette publications. One author's name stood from the rest; Jim Butterfield.

I used to turn to Jim's articles immediately when I managed to get my hands on a new magazine. Mr. Butterfield has the rare ability to

describe complex subjects in simple terms.

I'm certain that I'm not alone when I credit Jim with having taught me a lot about the inner workings of the Commodore 64. As important as the specifics of writing code for the C64 was Jim's style. He would often write code that was readily portable to multiple CBM machines. His code had longevity and purpose. The solidity of his programs left me with a lasting impression pertaining to how software should be developed.

The following interview with Jim was conducted via e-mail.

Q: What was the first programming language that you learned?

A: In about 1963, an assembly language called COGENT for a computer that few people have ever heard of: a Collins Radio C-8401. That was shortly followed by work on an IBM 1401, which had a machine language that was alphanumeric. (Honest! You could keypunch M/L directly!)

Q: Were numbers expressed in Base-36?

A: No. Decimal.

The basic machine had 1000 bytes (not 1K) of (7-bit) memory (core, not RAM!) so addresses ranged from 000 to 999 (and were given in decimal, of course). Expanded machines had 4K, then 16K ... the addresses were slightly more complex in that case.

Thus, to move bytes from an area at, say address 123 to address 456 the instruction would be M123456. I AM NOT MAKING THIS UP!!!!

Q: Did you guys have contests to spell out goofy words as part of a program? (I know of a programmer who used to regularly use the return code $0BAD to indicate a problem...)

A: No (the addresses mixed in with the op codes ruled that out), but you could do fun things on a 1401 if the system manager wasn't looking .. such as play music.

Q: What was the first computer that you owned?

A: Not counting the TUTAC-1, which was powered by rubber bands and was more correctly a logic machine: The KIM-1, a single-board microcomputer made by MOS Technologies, Inc., of Norristown PA. MOS Technologies was subsequently acquired by Commodore.

Q: When did you first encounter a Commodore computer?

A: When Commodore acquired MOS Technologies, the computer that I had owned for over a year became a Commodore computer. Subsequently, an employee of MOS Technologies, Chuck Peddle, convinced Jack Tramiel of Commodore that they should launch a personal computer called "The PET". I got one of those not long after they started production.

Q: Did you have formal training in computer programming?

A: Yes, on that long-ago Collins C-8401. But this was more a process-control machine; it didn't use of any the newfangled (at the time) languages such as Fortran and Cobol. So my training was in machine language/assembler.

Q: What was the first book that you wrote?

A: A couple of enthusiasts and I collaborated on a volume called "The First Book of KIM", a book describing how to do things with the KIM-1 single board computer. That computer was powered by a 6502,by the way; in fact the KIM-1 board itself was designed as a engineering prototype for people who wanted to try out the chip.

Q: Was it similar to the Altair where you had to manually increment an address-counter before you could throw the switches to set the byte at that address?

A: No, the KIM-1 had an operating system in ROM. That's one of the things that made all KIM users "equal" and able to share programs, while the other early micro owners had quite a scattering of stuff.

Q: What COULD you do with a KIM-1?

A: Hey, watch it! That's like saying, "What could you do with a Commodore 64"? Although the KIM-1 came with a hexadecimal keypad rather

than a keyboard, and output to a six-digit LED display, you could use those to good advantage AND hook up extra stuff. Play music? Play Blackjack? Hunt the Wumpus? Skeet shoot? Unless you had the budget for a printer, you'd have a hard time doing an accounts receivable, of course. But this is the 6502 we're talking about! And we all know it can do ANYTHING!

Q: What was the last book that you wrote?

A: It's probably the revised version of "Machine Language For the Commodore 64, 128, and Other Commodore Computers". In 1985 and 1986, however, I did produce a "pocket diary" reference guide for Commodore 8-bit computers.

Q: Have you ever written articles or books on subjects that are not computer-related?

A: My first writing experience was a treatise on transistor theory, published by Popular Electronics in August of 1959. Not much else.

Q: Did you write commercial software for any of the Commodore computers?

A: As a general rule, no. All my stuff is public domain. At one time, I had written a simple spell-checking engine that was incorporated into a word processing package for a while.

Q: SuperMon was a tool that I used daily when developing ML routines or exploring the C64. What prompted you to write SuperMon?

A: In the early days of Commodore personal computers, there were quite a few machine language monitors around. They were partly based on some publicly published code by Steve Wozniak (of Apple!), and partly based on the MOS Technology TIM monitor, from KIM-1 days.

Two variants of the basic monitor caught my eye: NewMon, which added several useful features to the basic Machine Language Monitor; and HiMon, which sited the monitor in upper memory where it wouldn't conflict with BASIC programs. I decided to put the two together and generate a self-relocating MLM. That was desirable in early PET/CBM days, where some computers would come with 8K RAM, some with 16K, and others with 32K; you couldn't assume where the top of memory would be.

In those days, almost every Commodore computer came with a small built-in MLM, and the first Supermon was an add-on. Later, as Commodore changed the style of the MLM packages they built into newer machines such as the 128, I went back and modified those earlier versions so that they would work the same across all platforms.

Q: Did you ever expand the mini-assembler in SuperMon into a full-blown assembler development package?

A: No. I hustled Brad Templeton into writing PAL, so that there would be an assembler available for those who needed it. There had been a few assemblers around before that - Commodore had one, and another was the MAE system - but I was sure that somebody like Brad could do better.

Q: Even Superman had to put up with Kryptonite. Describe your worst experience as a software developer / technical writer.

A: My first publication of SuperMon in Compute! magazine had the wrong end-of-address supplied (my fault). I got a LOT of mail and phone calls on that one.

Q: I had heard a rumor pertaining to your software development habits that indicated you would approach a given project with full force. You would focus your undivided attention on it until it was complete. Is this rumor accurate?

A: Possibly. If I have a project under way, it "follows me around" until it's complete; I fret over it and can't put it away until all the pieces are in place.

Q: If so, did you ever change this methodology?

A: Not to any great extent. A half-written program bugs me, and I won't rest until it's finished.

I might, however, decide that I'm taking the wrong track, and scrap a program completely in order to start over. This isn't a loss: the first attempt can show you what's really wanted.

Q: Your articles made you seem a bit omniscient. You always had the inside info on the newest CBM computers and always seemed to be able to explain their complexities in a manner that would suggest that you had a lot of time to study them. I don't know a whole lot about your employment during the mid/late 80's. Were you affiliated with CBM? A beta-tester?

A: I had many friends in Commodore Canada, but I never worked for the company, although I did contract work for them on occasion.

The big problem was not getting information from Commodore; it was learning to ignore most of it. Commodore was bubbling over with ideas and plans that never came to fruition. There was no point in writing about projects that never happened (the Commodore music box? the cash register? the videotape/disk storage device?). I took the position: "Don't tell me about it until it's a real product!".

Commodore Canada was an excellent source of information, and I relied on them to keep me from straying too far into technical speculation.

Q: Did you use any high-level languages on CBM computers?

A: BASIC, of course. COMAL, a BASIC derivative language from Denmark, was nicely constructed. Played around a little with C, but that language doesn't fit comfortably into an 8-bit environment.

Q: What was your favorite computer that CBM produced?

A: I don't know that I have a single favorite. The early PET/CBM machines were great "discovery" platforms, where we could investigate these wonderful new computers. The advent of the VIC-20 and the Commodore 64 brought color and sound, which added to the charm of these home computers; but they paid a penalty in slow disk access and screen width limitations. Today, perhaps the Commodore 128 ranks as the best, or at least the computer with most general usability. But it wasn't produced in quantities as great as some of the earlier machines, and so the user community hasn't been quite as furious.

Q: What kind of home computer do you currently use?

A: C128 .. Amiga .. Pentium system. All three.

Q: Who were your influences as related to writing?

A: Nobody specific. Just tried to write it as I would say it.

Q: Who were your influences as related to programming?

A: I've worked with a lot of sharp programmers over the years. Not one I can pick out especially.

Q: If you could relive the CBM glory years, would you do anything differently?

A: I don't think so. On another path, I could have gone for big bucks; but making money carries a responsibility to support and service, and that would have taken the fun out of it.

Q: Is your current job computer-related?

A: I'm currently more or less retired.

Q: If you had not chosen a career in computing, what field of endeavor would you most likely have pursued?

A: Before computers, I worked in electronics and telecommunications.

Q: What are your current hobbies?

A: Reading; travel; films; raising my daughter. (That's a hobby???)

Q: What sort of technical literature do you currently read?

A: Mostly reference material. Current magazines are heavy on the "what's for sale" stream; to my mind, that's not the fun part of computing.

Q: Are you surprised that a sort of "CBM renaissance" has been taking place the last few years (...availability of C64 emulators on multiple platforms and such...the SuperCPU from CMD...).

A: It's a shame that Commodore wasn't able to/interested in keeping the

8-bit line going. It's good to see that is happening.

Surprised? A little. But enthusiasts and user groups have always had a stronger effect than manufacturers are willing to admit.

Q: What is your opinion on the way consumer computing has evolved since the inception of the early PET machines?

A: The average computer user today has a lot less fun than we still have with the early machines. The industry message today is "Buy it and use it, and then turn it off .. don't worry or think about how it all works". That's sure a lot less fun for tinkerers.

Q: What words of wisdom would you care to impart on a new (or revitalized) generation of CBM hackers?

A: Enjoy what you're doing! If it becomes drudgery, you're doing it wrong!

==========

@(#)trivia: Commodore Trivia
by Jim Brain (j.brain@ieee.org)

@(A): Introduction

As some may know, these questions are part of a contest held each month on the Internet, in which the winner receives a donated prize. I encourage those who can received the newest editions of trivia to enter the contest.

This article contains the questions and answers for trivia editions #29-32, with questions for edition #33.

If you wish, you can subscribe to the trivia mailing list and receive the newest editions of the trivia via Internet email. To add your name to the list, please mail a message:

To: brain@mail.msen.com
Subject: MAILSERV
Body:
subscribe trivia Firstname Lastname
help
quit

@(A): Trivia Questions and Answers

This edition should be sub-titled "Programmer's Trivia".

Q $1C0) What are the two configurations for the LORAM, HIRAM, GAME, and EXROM pins that will allow the use of a full 64kB of RAM in the C64?

A $1C0) There are actually 4 configurations, in two categories:

LORAM	0	0	(X means either 1 or 0)
HIRAM	0	0	
GAME	1	X	
EXROM	X	0	

Q $1C1) What is the first thing that the C64 (and VIC) KERNAL does upon powerup?

A $1C1) The first thing each does is reset the stack pointer to $ff.

Q $1C2) What KERNAL routine is used to set a DOS channel to input?

A $1C2) CHKIN ($ffc6)

Q $1C3) What KERNAL routine is used to set a DOS channel to output?

A $1C3) CHKOUT ($ffc9)

Q $1C4) Before calling the routines in $1C2 and $1C3, what register must you load?

A $1C4) You must load .X with the logical file number.

Q $1C5) What 3 devices can the KERNAL NOT load from?

A $1C5) keyboard (0), RS-232 (2), or screen (3). The first and last are somewhat obvious, but allowing RS-232 loads would have made loading from a remote machine possible. Incidentally, you can't

save to any of these devices, either.

Q $1C6) In the Commodore KERNAL, there are "high" and "low" level routines. To which class of routines does "SECOND" belong?

A $1C6) low. It is used to specify the secondary address, as the '7' in open 4,4,7.

Q $1C7) If a programmer calls the KERNAL routine "STOP" and the RUN/STOP key is NOT pressed, what is returned in the .A register?

A $1C7) .A will contain a byte representing the last row of the keyboard scan.

Q $1C8) The Commodore KERNAL routines are all accessed via a jump table. What routine is used to change the values in the KERNAL jump table?

A $1C8) The appropriately named VECTOR ($ff8d) call, which few programmers actually use.

Q $1C9) A call is made to a KERNAL routine, the call returns with the C bit set and the .A register holds $02. What error does this indicate?

A $1C9) "File already open"

Q $1CA) If a call to READST is made, and a $40 is returned in .A, what does this indicate?

A $1CA) End of File.

Q $1CB) What routine can be called to determine the physical format of the Commodore 64 screen in characters?

A $1CB) The also appropriately named SCREEN ($ffed) call.

Q $1CC) The Commodore 64 starts a non-destructive RAM test at what location?

A $1CC) $0300.

Q $1CD) Which way does the RAM test proceed: up or down?

A $1CD) up.

Q $1CE) Which KERNAL routine is used ONLY in conjunction with a Commodore IEEE card?

A $1CE) SETTMO ($ffa2), which sets the IEEE bus card timeout flag. I infer that Commodore thought many people would use the IEEE interface. (Anyone know any more about this?)

Q $1CF) Many hybrid BASIC/ML programs use SYS to transfer control from BASIC to ML. However, a few use USR(X). When using the latter function, where does BASIC fetch the ML routine's starting address from?

A $1CF) 785 and 786, in classic LO:HI format.

The "BASIC" Trivia Set

Q $1D0) To load a program from the current location on a cassette tape, what two key combination must a user press on a VIC-20 or C64.

A $1D0) SHIFT and the RUN/STOP key. Note that the same key sequence loads a file from disk on the SX-64 or C128 in 128 mode.

Q $1D1) If I issue the BASIC statement OPEN "JIM,S,W", What type of file am I opening?

A $1D1) A sequential file.

Q $1D2) Is BASIC in the Commodore computer systems an "interpreted" or "compiled" language

A $1D2) interpreted. When a program has been "Blitzed!", it is then compiled.

Q $1D3) What type of variable is A%?

A $1D3) An integer variable.

Q $1D4) If I issue the BASIC line PRINT:PRINT "A","B" what column does

the "B" show up on when run on a C64?

A $1D4) Column 11, if we number columns from 1.

Q $1D5) What column does "B" show up on if I run the BASIC line in $1D4 on a VIC-20?

A $1D5) Column 12. Since the VIC has 22 columns, the natural column spacing was 11 positions, instead of 10 on 40 and 80 column CBMs.

Q $1D6) Alphebetically, what is the first BASIC 2.0 command to have a 3 letter abbreviation?

A $1D6) CLOSE.

Q $1D7) How many times does the statement FOR T=1TO0 execute?

A $1D7) once. A BASIC for loop always executes at least once. This is different from languages like 'C', which would not execute the loop at all. Feature or bug, who knows...

Q $1D8) What base does the BASIC LOG command use for its logarithm function?

A $1D8) base e. (2.7....) (one of the entrants claims that "e" in the 64 isn't quite as accurate as we think. He was quoting 2.85....

Q $1D9) A = NOT B can be written as which expression:

a) A = -B
b) A = -(B+1)

A $1D9) b. NOT computes the twos-complement of the number, not the simple ones-complement negation. This feature simpleifies subtraction in a CPU, since subtractions can be performed as additions.

Q $1DA) What does INT(-15.43) return?

A $1DA) -16. INT returns the next LOWER integer.

Q $1DB) What does ASC$("JIM") return?

A $1DB) ASC$ returns an error. That's what I get for writing these late at night. What I menat was "ASC", returns the value of the first character of a string, in this case 'J'. Since I didn't specify if this was a uppercase 'j' or lowercase 'J' in graphics mode, the result could either be 74 or 202.

Q $1DC) What is the abbreviation for GET#?

A $1DC) Technically, there is none. However, on the C128 at least, GET# shares the same token as GET, so typing gE# will indeed work. This is different from PRINT and PRINT#, which have different tokens.

Q $1DD) What is the largest integer value that Commodore BASIC can handle?

A $1DD) Again, this was a little ambiguous. I was looking for the maximum value that an integer variable can hold, which is 32767, but line numbers (which are integers) can be up to 63999.

Q $1DE) What is the ONLY Commodore Editor key not affected by "quote mode"

A $1DE) The DEL key. I would have answered return, but the 64 PRG spells it out that only this key is unaffected.

Q $1DF) What is the range of RND?

A $1DF) 0.0 <= RND < 1.0, or [0,1). Both mean that the range is from 0 to 1, including 0, but not 1.0.

The "VIC Chip" Trivia Set

Q $1E0) We all know that VIC stands for Video Interface Chip. However, in what computer was a VIC chip first used?

A $1E0) The VIC-I was used in the VIC-20.

Q $1E1) What is the difference between the 6566 and 6567 VIC chips?

A $1E1) The 6566 has fully decoded address lines. The '67 has multiplexed

address lines for connection to DRAM.

Q $1E2) On what computer would one find a VIC-II chip?

A $1E2) C64, C64C, 64SX.

Q $1E3) On what computer would one find a VIC-IIe chip?

A $1E3) C128, C128D

Q $1E4) On what computer would one find a VIC-III chip?

A $1E4) C65 (64DX)

Q $1E5) Versions of each VIC chip exist for each computer model/video standard combinations suppoerted by Commodore. What model/video standard would the 6569 work with?

A $1E5) C64 type machine using the PAL-B standard. Note that there are also PAL-N and PAL-M standards, which required different VIC-II models.

Q $1E6) How much memory could be directly addressed by a VIC-II chip?

A $1E6) 16 kilobytes.

Q $1E7) How many control registers does the VIC-I contain?

A $1E7) 16 control registers.

Q $1E8) How many control registers does the VIC-II contain?

A $1E8) 47 control registers.

Q $1E9) The VIC-II series introduced Movable Object Blocks to the Commodore programmer. By what common name are MOBs known?

A $1E9) "sprites"

Q $1EA) What are the dimensions of a MOB?

A $1EA) 24 dots wide by 21 tall.

Q $1EB) What difference between the VIC-I and VIC-II causes VIC-II equipped systems to potentially operate slightly slower than VIC-I equipped systems, all other items held constant?

A $1EB) Even with all of the fancy features of the VIC-II (like sprites) turned off, the VIC-II doesn't have quite enough time to do all of its work, which includes refreshing the DRAM ICs in addition to the work of drawing the screen and reading the Paddle inputs. So, every 8th rasterline, the VIC has to "steal cycles" from the CPU to fetch character data from RAM. Since this time is not available to the CPU to execute programs, a simple program written on each machine will execute faster on the VIC because its CPU doesn't have to fight the video IC for cycles.

Q $1EC) In addition to supporting graphical output to an external display, what other vitally important function do the VIC chips (starting with the VIC-II) perform?

A $1EC) They refresh the Dynamic RAM of the computer periodically. If the DRAM is not refreshed, it would lose its contents.

Q $1ED) Many people know that the VIC-II can deliver up to 320x200 resolution without much trouble. What is the maximum resolution of the VIC-III chip?

A $1ED) According to the specifications, it is supposed to handle 1280H by 400V interlaced and non-interlaced.

Q $1EE) Between the development of the VIC-II and the VIC-IIe, there was a related, though not very similar video IC developed for CBM machines. Name its TLA (three letter acronym).

A $1EE) TED (Text Editting Device). It was developed for the 264 series (Plus/4, C16).

Q $1EF) How many pins does a VIC-II chip contain?

A $1EF) Every VIC-II has 40 pins.

The "BASIC Tokens" Trivia Set

The following questions refer to the way Commodore "crunched" BASIC programs by substituting one of more bytes called "tokens" for BASIC keywords in a BASIC program. The resulting code was smaller, since multiple character keywords were internally replaced with smaller length tokens.

(All the answers were taken from _Commodore Magazine_, April 1987, pp 82-85.)

Q $1F0) Commodore BASIC tokens start at what number?

A $1F0) $80, or 128.

Q $1F1) BASIC 2.0 defines tokens without gaps up to $ca. What keyword is represented by $cb?

A $1F1) GO.

Q $1F2) Why is the token for PI strange?

A $1F2) It is token $ff, or 255.

Q $1F3) All versions of Commodore BASIC contain at least a subset of tokens. At what number does this subset end?

A $1F3) $ca.

Q $1F4) BASIC 4.0 defines tokens beyond $cb. What is the last token included in BASIC 4.0?

A $1F4) $da.

Q $1F5) There was a BASIC 4.0+ included in the B series. It extends the BASIC with some new commands not in 4.0. What token range are these new commands at?

A $1F5) $db-$e8.

Q $1F6) When a user plugs a Super Expander into a Commodore 64, he or she gains access to 25 new BASIC commands. The tokens for these commands are defined differently from the previous tokens. What is the difference?

A $1F6) They are two byte tokens of the form: $fe XX, where XX ranges from $80 to $9e.

Q $1F7) When the Plus/4 and C-16 was developed, new commands were added to BASIC. In addition, many commands from BASIC 4.0 were also included. Unfortunately, the tokens for BASIC 4.0 commands included in these new machines differed from those in the older BASIC 4.0. If a user lists a program written in BASIC 4.0 on a Plus/4, what will the BASIC 4.0 CONCAT command show up as?

A $1F7) CONCAT is $cc in BASIC 4.0, and is RGR in BASIC 3.5.

Q $1F8) What is the last token used in the Plus/4 line?

A $1F8) $fd.

Q $1F9) If you list a program written on the Plus/4 with the keyword SCALE on a BASIC 4.0/4.0+ machine, what happens?

A $1F9) SCALE on BASIC 3.5 is token $e9, which is not in the BASIC 4.0(+) list. The PET will crash. Interstingly, tokens above $e9 do not crash the PET.

Q $1FA) When the C128 was released, it shared many tokens with the Plus/4. However, at $ce, the 128 differs from the Plus/4. The Plus/4 token $ce corresponds to RLUM, but the C128 uses the token another way. What is peculiar about the C128 usage?

A $1FA) The C128 uses $ce as a prefix byte for a range of two-byte tokens that range from $02 to $0a.

Q $1FB) The C128 shares many keywords with the Super Expander cartridge for the C64. As with the Plus/4, though, keywords don't map to the same token. To what token does the C128 keyword SPRITE (token: $fe $07) correspond to on the Super Expander equipped 64?

A $1FB) $fe $93.

Q $1FC) What keyword was not included in BASIC v1, but was included in
BASIC v2?

A $1FC) GO, token $cb.

Q $1FD) The C128 defines all the tokens from $fe $02 to $fe $26, with the
exception of two tokens. Name one of them.

A $1FD) $fe $20 and $fe $22.

Q $1FE) The Plus/4 line had the ability to add keywords dynamically when
running cartridges. At what point in the token list do these
"added" keywords show up in the Plus/4 line?

A $1FE) They use $fe as a prefix byte for two-byte tokens.

Q $1FF) If a programmer want to write a single program to run on a
B128, a plus/4, and a C128, what version of BASIC is the lowest
common denominator?

A $1FF) Unfortunately, BASIC 2.0 is it.

The C128 Set:

Q $200) How many general purpose central processin units does a C128
contain?

Q $201) The Commodore 128 contains a MMU IC. What does MMU stand for?

Q $202) What Commodore produced cartridge is specifically mentioned in
the 128 PRG as being incompatible with the 128?

Q $203) The C128 introduces the concepts of "banks" How many such banks
are recognized by the C128 BASIC?

Q $204) What version is the BASIC included in the C128 in native mode?

Q $205) Can any of the BASIC graphics commands be used on the 80 column
screen?

Q $206) How many high-level graphics commands are available on the C128
in C128 mode?

Q $207) In C128 mode, at what location does screen memory start?

Q $208) The 80 column IC in the 128 can display how many full character
sets of 256 characters each at one time?

Q $209) Many have scorned the C128's 80 column video IC. What about this
IC makes it so hard to use?

Q $20A) What number is the 80 column IC referenced by?

Q $20B) What machine language addressing modes cannot be used with the
80 column chip?

Q $20C) The C128 contains keyboard keys not present on the C64. What IC
is used to read these keys? (besides the CIA, as on the 64)

Q $20D) Following the introduction of the C128, a new version of was
developed. Name it.

Q $20E) Many people refer to C128s as 16k or 64k units. To what does this
refer?

Q $20F) According to the C128 literature, the C128 can be expanded to use
how much memory?

==========

@(#)basic: Hacking BASICs
by R. T. Cunningham (wanderer_rtc@usa.pipeline.com)

@(A): Introducation

Although BASIC is not an "advanced" language, such as Assembly or C, it's
been used far more often than any other since the first Commodore computer
was introduced. It is not my intent to provide information here that is
widely available in user manuals, newsletters, or other forums. What I am

going to present is what I have learned from "The School of Hard Knocks" or, in other words, by trial and error.

@(A): Detecting Drives Connected

If your program attempts to access a device number that is not connected to computer in the serial chain, like fetching a directory, your program will halt with "?device not present error in (line number)". How do we prevent this from happening? The answer is to open the device number, close the device number, and check the status variable:

```
open15,10,15:close15:ifst<>0then (act on device not being present)
```

Any number returned by the status variable other than 0 will indicate that the device is either not connected or not turned on.

In my own programs, I like to do this check for all of the drives early in the program by stashing the device numbers in an array. Legal device numbers for disk drives are 8 to 30, with 30 being used as configuration mode for a CMD drive. Since I know of no other drive that can use device #30, I won't include that device number in the loop. The array should be dimensioned to at least 21. I prefer 22 and to use the first variable (0) to hold the number of devices. I keep the "start up" device number in a non-array variable so that I can return to it after the drive checking routine has been completed:

```
10 dimdv(22):dv(0)=0:rem set number of devices to 0
20 dv=peek(186):rem current device # of last drive accessed
30 ifdv<8thendv=8:rem prevents non-disk device #s
40 fora=1to22:rem device #s 8-29 minus 7 so that array starts at 1
50 open15,a+7,15:close15
60 ifst<>thendv(0)=dv(0)+1:dv(dv(0))=a+7:rem increment number of devices
and store device numbers
70 next
```

If I have device number 8, 10, 12 and 14 connected, dv(0) will now contain 4, dv(1) through dv(4) will contain those device numbers. You can now use this array to check for a valid device number prior to an access command:

```
100 rem d equals the device number selected in this example
110 fl=0:rem set flag to 0
120 fora=1todv(0):rem check total number of drives attached
130 ifd=(dv(a))thenfl=1:rem set flag to 1 if a match is found
140 next
150 rem continue with drive access only if flag is set to 1
```

@(A): Selective GOTO (GO TO) Routine:

When acting on a number the ON/GOTO command set works by either using a numeric variable or the value of a string, if the string contains a number:

```
10 onagoto100,200,300,400
```

or

```
10 onval(a$)goto100,200,300,400
```

What if we want to use a non-numeric character with ON/GOTO, especially with a list of choices presented to the user?

In the C128's native mode the INSTR function can be used:

```
10 rem keys accessible by user are A,B,C,D and stored in a$ response
20 oninstr("ABCD",a$)goto100,200,300,400
```

or

```
20 a=instr("ABCD",a$):onagoto100,200,300,400
```

On the 64, since INSTR is not an available function, we can emulate the function with a different routine:

```
20 on-1*(a$="A")-2*(a$="B")-3*(a$="C")-3*(a$="D")goto100,200,300,400
```

or

```
20 a=-1*(a$="A")-2*(a$="B")-3*(a$="C")-3*(a$="D"):onagoto100,200,300,400
```

Note that the first 1* is not necessary but the - is. I've added it for clarity only.

In the first example, INSTR returns the place number (sequence) of each character in the string. In the second example, the place number is obtained by multiplying the negative sequence number by the signed value of a$. The signed value of a string is always -1 (I think!).

When working with a program that is designed to work in both 64 and 128 modes, the INSTR function should not be used since the other method will also work in 128 mode.

More proficient readers might want to demonstrate the machine language equivalents of both drive detection and INSTR routines? I for one would like to add them to my ML arsenal.

==========

@(#)error: ? DS, DS$: rem The Error Channel

We are not aware of any errors with issue 13.

==========

@(#)bits: Twiddling the Bits: VIC-20 ROM Cartridge Exploration and Archiving
by Ward Shrake (wshrake@aol.com)

@(A): Introduction

This article's primary purpose is to enable you to understand the basic principles of making an archival backup of a ROM cartridge. However, I do wish to point out the following important points, before I begin:

@(A): Disclaimer

1. This is NOT intended for any sort of illegal purposes! The information can be misused, if one wants to badly enough. However, so can virtually every other piece of information on the planet. Fire is a good thing, as are hammers, saws, and other tools, but all of them can be misused. I urge the reader to use this information in a proper fashion and to take the time to reflect on what you are doing, to avoid hurting anyone or anything.

2. I am releasing this information for two basic reasons (besides basic hacker pride in obscure technical knowledge). First, the Vic20 died many years ago. Commodore helped accelerate its death, by pushing the C64 computer onto the market, at a time when all the companies out there had a lot at stake. I don't blame Commodore from a marketing sense; they just wanted to stay alive themselves in a cutthroat market environment. However, this means a lot of people still have the idea that the Vic20 was/is a piece of obsolete junk. This is not true, but the public acts as if it were. The end result is that a lot of perfectly good hardware and software is thrown away on a regular basis. Once it is in a landfill somewhere, it’s rather difficult for anyone to use it! Hence my two-fold concern: to help show the public (and Commodore Guru types) the Vic20 is a very cool gaming machine, and to physically rescue the software for the system before all of it is lost forever. I take little pleasure in thinking that 50,000 years from now some archaeologist will dig up our landfills and think we must have had some really cool stuff!

3. I really like the term "Digital Archaeology." (I wasn't the first to use it, however.) Basically, it means that with the many rapid advances in the computer sciences, lots of stuff gets lost or forgotten in the rush to replace everything every few years. Whereas an entire civilization might take hundreds or even thousands of years to die out and be all but forgotten, the history, lore, and software of computers can be gone in just a few years. If we don't revive it now, who will remember how to use any of this stuff? There are only a few people around now that can do it, and their memories are getting fuzzy with the passage of time. I don't mean to preach; I just think this is as good a time as ever!

A small group of concerned individuals, myself included, have begun the task of archiving all the VIC-20 cartridges known to exist. The project also includes documenting every aspect of the VIC-20 hardware and how it operates.

@(A): General information about Vic20 cartridge software

A Vic20 cartridge is approximately 5.5 inches wide, 3 and 3/8ths inches deep, and 5/8ths inches tall, when viewed as if ready to be installed. Most of the cartridges that were made by Commodore are either a beige or dark brown colored plastic with a tan or metallic all-text label. Some third-party companies had more interesting looking cartridges; most of

the plastic cases were black as a general rule, with white also being used at times. In rare cases, oddly-shaped carts existed, for example, carts by UMI. Some of these seem to be more epoxy-based than plastic, with glitter inside it! Only a few companies really had fancy, full-colored labels on their cartridges.

The most common memory configuration, inside the casing, is one bank of 8k. ROM was standard for the larger companies, but EPROM's were also used at times, even by Commodore themselves. Four-kilobyte cartridges do exist, as do 8k cartridges made up of two banks of 4k chips. (Memory was expensive then, as you can no doubt imagine, and some companies had to make due with whatever they could get cheap enough.) 16K cartridges do exist as well, although the bulk of these were done by the largest companies. Commodore made a few 16k carts, as did Atari, Sega, Hes, and a few others.

If other standard cartridge memory configurations existed in the Vic's day, the author is currently unaware of them. (I would like to hear of any, if they do/did exist, especially if they were once made commercially.) There may be a few exceptions, for instance, for carts that modified the Vic20's own inherent abilities; for example, 40 and 80 column boards. But the rule of thumb here is that 8k was the accepted standard, with 4k and 16k at times. All of these cartridge configurations, by the way, are using 8-bit memory because the Vic20 is an 8-bit computer.

Inside a typical Vic20 cartridge is one double-sided, etched circuit board, with some form of memory chip(s) on it. This may be a "standard" IC chip as we are used to seeing (24 or 28 pin ROM, in a DIP package), or it could be a blank circuit board, with a tiny blob of black epoxy material on it, under which are presumably the internal components of a typical, normal ROM chip.

The standard circuit board size is approximately 3 and 9/16ths inches wide, and 1 and 3/4 inches deep. A cart fits into a 44-position, double-sided card edge connector, located in the back, left hand side of the Vic20.

On the cart circuit board itself, there might be several wire traces or jumpers, which, if there, are meant to configure the cartridge to a certain memory arrangement. These jumpers are meant to connect the BLK or RAM lines to the system ground. These lines, when connected, tell the Vic20 where to place the cartridge within the Vic20's internal memory mapping scheme.

While all "normal," autostart game cartridges are located in one fixed area of memory, this is not true with all cartridges. A rare few of the total Vic20 cart collection did not use the autostart procedure; you had to type in a systems number to start the program running after inserting it. These are very rare, however (6 out of 150+ so far). Most cartridges all autostart after insertion and power-up, just as cartridges do in game console systems.

All this memory banking may seem confusing. If it does, remember this: Every cartridge that autostarts must have at least one bank of memory in Block 5, or the autostart procedure will not function. If it autostarts, there has to be some memory in Block 5. That may help relieve some confusion. There may be additional memory installed as well, in some cases, for more storage space. The location of this additional memory is less important than the location of the autostart bank. However, there are only three additional 8k banks left, for a total of four banks of user-added RAM or ROM memory. The Vic20 was built when memory was expensive, and computers were designed to be bare-bones memory-wise with the capability to expand on later. This may take some getting used to at first, but understanding it gets easier in time. Take it in stride for now.

@(A): The Cartridge Auto-Starting Feature

Please note that if you already understand the autostarting system used in the Commodore 64, this is nearly identical in procedure. Only certain codes and memory locations differ; the rest applies to both machines.

The "normal" spot for an 8K cartridge to be located in memory is in "Block 5"(which is located at $A000 to $BFFF). Again, this is not the only possible spot in memory for a cartridge to be located, but it is the only spot where the Vic20 will look for a cartridge that automatically starts on power-up. This is because (at power up) the Vic20 looks for a precise code in a precise spot to see if it should autostart a cartridge or not. If the Vic20 finds this EXACT five-byte code, EXACTLY where it is supposed to be located in memory, the Vic20 turns control over to the

cartridge. If not, it gives over control to the user, via the normal, power-up Basic READY prompt screen.

This 8K autostart sequence code is shown below:

Address	Hex value	Decimal value	ASCII values
$A004	$41	65	Capitol "A"
$A005	$30	48	Digit "zero"
$A006	$C3	195	Reverse "C" character
$A007	$C2	194	Reverse "B" character
$A008	$CD	205	Reverse "M" character

(You may note that the code above is symbolically saying $A000, and Commodore Business Machines. In other words, that this is the right software, for the right machine. The C64 uses $8000 instead.)

If the computer finds this five-byte sequence exactly as shown, it turns its control over to the machine language program in the cartridge. To do so, it needs to know where the program begins. There are four bytes which determine this, as shown in the chart below. Note that this is in the cart, not the Vic20.

Address	What this byte of information contains
$A000	Low-byte of 16-bit "cold start" address (power-up)
$A001	High-byte of 16-bit "cold start" address
$A002	Low-byte of 16-bit "warm start" address (restore key)
$A003	High-byte of 16-bit "warm start" address

Let's do a quick summary of this before we go on. You plug a cartridge into the Vic20. You turn the power on. The Vic20 starts its own built-in operating system software. It gets itself ready to be used, either by the user (in BASIC) or by a cartridge's program.

One of the last steps in the computer's start-up sequence is to look at a certain spot in memory, to see (A) if there is a cartridge inserted there, and (B) if it is the proper type. It does both these tasks through simple assumptions. Assuming that the existence of an autostart code sequence at a certain memory address implies a cartridge is present, the computer looks for the "A0CBM" code sequence. If it finds it, it then transfers control to the locations specified at $A000-$A003, as defined above. (In many cases, this address is $A009 or 40969 ... the next byte possible after all the start-up codes.)

@(A): How to Archive Vic20 ROM Cartridges

Now that we know how the autostart process works, what can we do with that information? Well, to archive a cartridge's internal ROM memory to either tape or diskette, you have to know how the autostart process works. This is necessary because you have to figure some way around it to be able to get to that "no carts inserted" normal power-up screen. This means that you then have full control of the machine, instead of the cartridge being in full control.

Simply put, there are only two steps to archiving a cartridge from its cart to disk or tape.

1. Find some method of defeating the autostart process, so that the Vic20 powers up with the cartridge in memory, but does not start it up. So that you are in control, instead of the cartridge running the show.

2. Copy that area in memory where the cartridge resides, to tape or disk. The resulting program is called an "image" of the cartridge's memory.

After that, assuming there is no copy-protection coded into the original, you have a working version of the program stored on disk/tape instead of on ROM.

Some problems may arise, but don't worry, they are easy to get around if you know how to do it.

A. There may be more than one block of memory to copy. The one found at Block 5 is easy, because if the cart auto-starts, there must be memory that needs to be copied. However, additional memory may be present. You will need to determine where it is, and how much of it there is.

B. Depending on how you transferred the image to disk (assume I mean "disk or tape" from now on), was the loading address information stored

with the image? If it was you can reload it easily next time. Without those two important address bytes (these are separate from those we discussed earlier), the software won't load properly, and so will not work.

OK, so now you're convinced it’s impossible, right? For every problem, there is always a number of solutions. I wrote a small program that eliminates most of the problems inherent in archiving carts, and have figured out ways of eliminating most of the hassles. The program also makes the process more reliable, since archiving rare and cool things is the main idea. I'll make that program publicly available, soon, after a bit of polishing.

But you still have to get around that auto-start procedure. Without doing that first, you are dead in the water before you've even started. So, here's how you get around the autostart procedure. After that, its all downhill!

There are a number of suggested methods shown below. My suggestion is to pick out the method you like best and stick with it. The rest will just be to satisfy your built-in hacking curiosity, OK? Other methods exist but I didn't feel they warranted the space here. These are some of the best ways.

@(A)method1: Inserting a Cartridge Into a Powered Computer

Please pay attention to what I am about to say: this is the only method I DO NOT recommend! I mention it mainly because it represents a very real risk of killing your hard-to-replace Vic20 computer system. This is supposed to be an exercise in keeping things alive and usable, not killing more off! If you feel you want to try this, you do it at your own risk!

I once knew a guy that wanted to archive Vic20 carts, but wasn't willing to go to a lot of effort to find a good way to get around the autostart feature. When he told me what he was doing, I tried to warn him. He kept on doing it, saying he'd "just be careful." The next thing I know, he is asking me how to repair a computer that died suddenly. I told him the truth: (A) you don't, you go find and buy another one, and (B) you listen to me the next time I tell you that you're risking your almost irreplaceable hardware.

If that didn't convince you, well, I tried. There are better methods, by far! If you can't find a better one than this, pay someone to modify your Vic20, or just loan the cart in question to someone who is more used to doing this!

@(A)method2: Altering Your Computer's Operating System ROM

What this method does is change the copy of the code to check for, inside the Vic20, so that no cartridge's unaltered code ever matches it. So no autostart.

No cart will ever match the modified start-up code in your new operating system, so none will ever be seen as a cart. However, having one of these chips installed permanently could be a bad idea, as you cannot test or start your saved images quite as easily. The image won't autostart either, unless you are willing to keep swapping chips back and forth, or unless you know how to decode the starting addresses and type in systems numbers.

This is an elegant way to solve the problem, providing you possess tools to read and program EPROM ICs and can obtain replacement ICs that are pin-compatible. Thus, the elegant solution requires a substantial amount of resources.

However, one chip does exist, which is expensive if you buy it new, and may be hard to find as well. (Try posting to "comp.sys.cbm.") If you can find a Motorola 68764 chip, you're almost there! Just copy the Kernal rom to disk, modify the "A0CBM" code found there to be anything but that, and burn a new EPROM of your modified Kernal rom. The manual to the Promenade EPROM-writing machine makes programming these chips a snap. And the Promenade is still available today, from it makers, Jason-Ranheim Co. (Phone: 916-878-0785, or 1-800-421-7731.) These same 68764 chips work with C64's internals, too.

This does work. It has the advantage of requiring no modifications to your Vic20 which can't be easily reversed. Before I got tired of swapping Kernal chips back and forth, this was my preferred method of archiving cartridges.

@(A)method3: Using a Cartridge Port Epander

I've come to feel that a plug-in board is the best way to modify things just long enough to get past the start-up process, skip the autoboot phase, and get down to business. If you can find one, buy it! It makes life so much easier and nicer, in more ways than just this one. I highly recommend buying one.

Basically, this device lets you plug in lots of carts at once, and with just a button press, decide which one to use at any given time. Well, that switch is all you need! Just (A) turn the computer off and insert a cartridge, (B) make sure all the switches are de-activated, (C) turn the computer on. You should now be looking at a normal, BASIC power-up screen. Now (D) press the button that activates the slot that your cartridge is plugged into, so that the cartridge is now mapped into memory properly, although belatedly, and (E) follow the instructions in the next part of this text, to archive the cart.

Note that you can modify your Vic20 to do the same thing if you are handy with a soldering iron. But as I don't want anyone killing their only Vic20, maybe I'll save that for another article ... this one is almost at deadline, and I don't want to rush things and make any mistakes!(Besides, I don't know yet if anyone is interested enough to modify their Vic20's to do this??)

@(A): Saving One or More Blocks of cartridge Memory to Disk or Tape

Once you've defeated the auto-start feature of the Vic20, the memory contained in the cartridge is just another block of memory as far as the Vic20 is concerned. You have full access privileges, with all that goes with it, including the ability to save it to external storage.

You have three basic choices to save the block of memory to tape/disk. You can either (A) wait until I release my program that does it automatically, or (B) use a "memory save" feature of a machine language monitor program (not built in) to save the 8K block of ROM memory, or you can (C) just change four POKE's in the Vic20's memory. With this, your cartridge is seen as if it were in the BASIC program memory area so that the built-in "SAVE" command works.

It really works. The beauty of it all is that no additional programs are needed. To save any block of cartridge-usable memory in the Vic20, just type in four easy POKE commands, then tell the Vic to SAVE the program as it would normally save any BASIC program. (You can even copy the system ROMs that way, if you want to.)

For those of you already familiar enough with Commodore's style and memory arrangement schemes to make sense of this, bytes 43 and 44 (decimal, not hex) are the pointers to the Start of Basic memory area, and bytes 45 and 46 are the pointers to the Start of Variables, or in other words, the End of Basic. Follow this chart, to save a block of memory from these areas via pokes:

Block #	Hex Address	Poke 43,x	Poke 44,x	Poke 45,x	Poke 46,x
5	$a000-bfff	x = 0	x = 160	x = 0	x = 192
3	$6000-7fff	x = 0	x = 96	x = 0	x = 128
2	$4000-5fff	x = 0	x = 64	x = 0	x = 96
1	$2000-3fff	x = 0	x = 32	x = 0	x = 64

Here's an example: to save block 5 (the most commonly used memory block) you do the following steps.

0. Follow the previous instructions, to get to this point. (The cartridge has been inserted, power is now turned on, and the autostart has been defeated. The Vic20 is showing you its normal BASIC power-up screen.)

1. Type the following, hitting RETURN after each line. (Type carefully!)

```
POKE 43,0                  (and return)
POKE 44,160                (and return)
POKE 45,0                  (and return)
POKE 46,192                (and return)
SAVE "FILENAME.EXT",8      (and return)
```

2. Wait for the disk drive to finish its saving process. (Be patient.)

3. If you have more 8k blocks to copy, repeat all the previous steps, but adjust the POKE statements, according to the information in the chart. Note that the pokes to 43 and 45 are always zero; even page

increments. Also, note that the computer will get confused or lock up if you try to copy another block of memory without starting completely over.

4. When the busy light turns off again, turn the Vic20's power back off. Remove the cartridge, and if you have a modified 8k or 16k RAM expander insert it or activate it. If you have a 32k RAM expander, you can use it.

5. Load the disk's directory up, (or rewind the tape), and see if the file has been properly saved to the disk. It should show a file size of 33 blocks, if all has gone well ... 32 blocks x 256 bytes each = 8k, plus one block for the disk drive's overhead and the file loading address.

6. If everything looks fine so far, just load the newly created "image" into RAM memory, to see if it runs. Remember a few tips: as with the C64, you always have to use the ,8,1 loading conventions for any machine language program to reload into memory back where it came from.

   Also, to properly load up an image that has more than one 8k bank, you will have to type NEW after each load to reset some memory pointers. Otherwise, it moves BASIC to where the cart now is and gets confused. So multi-loads are: (A) LOAD"file1",8,1 (B) NEW (C) LOAD"file2",8,1

7. The last step is starting up the image, to see it in all its new glory! To do this, all you have to do is (A) press the reset button you've installed yourself, or if you don't have one installed, (B) type the following "reset" command into the computer... SYS 64802 (and return).

8. At this point, your image is probably running. If it is not, carefully recheck all the previous instructions. You may have made a mistake or two there, somewhere. (One mistake I have often made is to forget to re-activate the expansion chassis' slot, and essentially save empty air.)

@(A): Troubleshooting

But if you recheck everything, and it STILL doesn't work, try archiving another cartridge. If the first does not work, but the second one does, you have likely found one of the few protected carts out there. How to "break" the copy protection is way out of the scope of this article, so I can't help you there! Sorry. But usually, everything will be fine by now, if you've followed the instructions carefully, step-by-step. Only about 10% of the VIC-20 cartridges were copy protected.

==========

@(#)next: The Next Hack

What? Why are you reading this? Aren't you happy with the issue you have? We walked through the snow with no shoes uphill both ways to deliver this to you and you are still asking for more? Have you even read the issue you have in your hot little hands? Demanding reader, aren't you. OK, here's what's coming up:

- We continually receive comments from folks trying to learn ML programming. They have copies of C=Hacking in hand, but lament that the concepts are too advanced for them. Well, next time we'll review some resources for the beginning ML programmer. Two of them, _Coder's World_ and _Bonkers_ are organized in publication format and go over many of the details that every ML programmer must learn.

- We're not sure which one will get here first, but Frank Kontros is writing up some of his impressive hardware and software projects as we write. It might be the EPROM programmer, the Digital I/O board, or one of his many software exploits. WE'll spotlight one next time.

- For many years, Commodore 64/128 users have been able to purchase stereo SID cartridges like the SID Symphony by CMD. However, there's not an overabundance of games or applications that take advantage of the second SID. Frank Kontros overcomes that problem in a non-obvious way by showing how to build a "pseudo'stereo" adaptor for your C64 or 128. The effect is noticeable and it requires no programming changes.

- And, of course, C=Hacking's regular goodies.

Now, go back and re-read those articles. We need some sleep....

==========

@(#)code: Hacking the Code

Being a technical, developer oriented magazine, some articles featured in C=H include executables or other binary files as part of the article. All such binary files are included on the soft copy of this issue in this section. In an effort to retain the integrity of such binary files through distribution over various computer networks, the binaries in this section have been encoded using the UUcode format, a popular Internet binary-to-readable text encoding method. In order to execute or otherwise utilize these binary files, one must feed this section of the magazine to a UUdecoding application. Typical examples include UUXFER for the 64, uudecode on the ACE OS for the 64 and 128, and uudecode on most UNIX OS machines. Some encoders can decode multiple files, while others will require the user to manually split this section into individual pieces prior to decoding.

In addition to this section, there are other ways to retrieve the binary files featured in this issue. For those with World Wide Web access, the files are available at http://www.msen.com/~brain/pub/ To retrieve "dim4.lnx", simply access the URL:

http://www.msen.com/~brain/pub/dim4.lnx

For those with electronic mail access only, the Commodore Hacking MAILSERV server also contains a copy of these files. To retrieve a copy of "dim4.lnx", send the following email message:

To: brain@mail.msen.com
Subject: MAILSERV
Body of Message:

send dim4.lnx
help
quit

For some articles published in Commodore, the author or authors may also have other methods for accessing files mentioned in the article. These methods are described in the respective article.

Commodore Hacking always attempts to provide the reader with as many options as possible to retrieve uncorrupted binary files. Although none of these above methods is foolproof, the added redundancy helps overcome any shortcomings.

WARNING: The UUCode format translates files from binary to ASCII, not PETSCII. Therefore, either decode this section before downloading this section to a PETSCII mode computer system, or download this section without translation to PETSCII. Some decoder programs can handle PETSCII converted UUCode files, but the practice is not recommended because conversion is typically done in a telecommunications program and accuracy in translation cannot be guaranteed.

@(A)menucode1: Menu Toolbox at $1000 (4096)

```
begin 644 mbox1000.bin
M`!!,>1M,]!M,&QU,0!U,$1M,WR9,^1U,;AY,XQY,)R%,%1],0Q],!B=,AB!,
M,2%,WR5,=BA,?2A,"2M,%BM,'2M,TQ%,SQ!,3!%,O11,$A-,<1-,.Q-,6!-,
ME!-,G!-,^Q-,/11,FQ1,'!M,+1M,/AM,3QM,RQ1,]A1,!Q5,&!5,(15,*!5,
M7!5,>!5,B!5,K15,.A9,P!9,Q!9,!Q=,$!=,%Q=,&A=,;A=,@A=,BQ=,EA=,
MO1=,PQ=,R1=,TA=,YQ=,]A=,$1A,)AE,3QA,[QHX(/#_CLHLC,DL(-T;A/R8
MP`2P`6`8:02%_JD`A?V%^ZD3(-+_(`4?>*D`A0&B`Z``L=&1^\8!L?/F`9']
MR-#QYM+F_.;TYO[*$.;&_J#IQ@&M(-#F`9']R,8!K2'0Y@&1_<BMRBR1_<BM
MR2R1_:EWA0%8KLHLK,DL&$SP_ZD`C:HL(-T;A/R8&&D$A?ZI`(7]A?NI$R#2
M_R`%'WBI`(4!H@.@`+'1C5`L+*HL$`FM4"PIO\D@T`RQ^Y'1L?W&`9'SY@'(
MT-[FTN;\YO3F_LH0T\;^+*HL,!^@Z;']Q@&-(-#(Y@&Q_<8!C2'0Y@'(L?V%
MULBQ_873J7>%`:D`C:HL6$QLY:D`A?V%_HU0+(W$+"#=&YBL4"R9HRSN4"S`
M!=#O(*@=R1F0`JD9C<4LH`"Q(IEF+,C,Q2S0]2"]%*ZC+*``C+LL&"#P_R`%
M'ZVH+(V\+"#<$B#D__#[H@"-Q"RNNRR.4"S)$?`0R9'P'R#U$LD-\"_0`$PD
M$NZ[+*V[+,VF+-!2J0"-NRQ,I1+.NRRMNRS)__`#3*42K*8LB(R[+$RE$JW$
M+/`&K5`L3(`2K;LL&&VC+*J@`"#P_R`%'ZVG+(V\+"#<$JV[+!AI`87]J0"%
M_DPJ(*VC+!AMNRRJH``8(/#_(`4?K:@LC;PL(-P2K:,L&&U0+*J@`!@@\/\@
M!1^MIRR-O"P@W!),)!*LI"RMO"R1\S`&L=%)@)'1R,RE+/#MD.M@JIA(H`"*
MV68LT`B,NRRB#8[$+,C,Q2S0[6BHBF`@O10@W1N$TR#=&X36(&SE(*@=C5`L
MK5`L\`V@`+$B(-+_R,Q0+-#U8""H'<D`\!6-4"R@`+$BR4&0!`F`D2+(S%`L
MT/!@(*@=C5`LR0#P#J``L2(I?Y$BR,Q0+-#T8"#=&YBJH``8(/#_(*@=C5`L
M2HW$+*D4..W$+(73(&SE3"@3J?^-JBQ,41$@W1N,QBP@W1N,QRP@W1N,R"P@
MO12I$R#2_R`%'ZD8A?N@)['1S<8LT`RMQRR1T:W(+#`"D?.($.K&^S`=J2@8
M9=&%T:D`9=*%TABI*&7SA?.I`&7TA?1,O1.I$TS2_R#=&XS++"#=&XS,+""]
M%*D3(-+_(`4?J1B%^Z`GL?,I#\W++-`%K<PLD?.($._&^S#)&*DH9?.%\ZD`
M9?2%]$P6%"#=&XQ<+"#=&XQ=+"#=&XQ>+"#=&XQ?+""]%*762*732*Y>+*``
```

```
M&"#P_R`%'\Y<+*Q=++'S*0^JO:LLD?.(S%PLT/"EULU?+/`+YM8@;.4@!1],
M;11HA=-HA=9,;.4@J!V%^R#D__#[H`#1(M`+R)B%_:D`A?Y,*B#(Q/O0[/#C
MH!BYV0`)@)G9`(@0]6!(J0&-S2QH("D8JJD!H``@NO\@1QB%%"!'&(45IA2D
M%:D`(-7_H`"8D:Y@AON$_*`#L?N9!RR($/A,_!V&^X3\H`.Q^YD'+(@0^$QQ
M'HX,+(P-+$SO'HX]+(P^+&`@&B".%RR,&"R-&2R&^X3\&&7[A?NI`&7\A?R@
M`+'[C54L($<8C58L($<8C1HL("(@3#`?CE4LC%8LJ0&-S2RI`(T/+(U7+(U8
M+(U9+$Q?'X;[A/RD^Z7\H@&.S2Q,"2=(J0&-S2QH("D8KATLJ0&@``""Z_R!'
M&(44($<8A16F%*053*(@J0&-S2R&^X3\H``@&B"I`(T>+(T/++'[C4HLC1\L
MC0<L($<8C4LLC0@LC2`L($<8C4PLC0DLC2$L($<8C4TLC0HLC2(L($<8C48L
M($<8C4<LC20LC0TL($<8C4@L($<8C4DL($<8C5<LC4$L($<8C5@LC4(L($<8
MC4,L($<8C40L($<8C44L("(@3*@AAON$_"`:(*D!C1XLJ0"-#RR@`+'[C4HL
MC1\LC0<L&&D=C4LLC0@LC2`L($<8C4PLC0DLC2$L($<8C4TLC0HLC2(L($<8
MC48L($<8C4<LC20LC0TL($<8C4@L($<8C4DLJ0"-5RR-02R-6"R-0BRN/RRM
M0"R.0RR-1"P@1QB-12P@(B!,J"&-'BQ@AON$_*D!C<TLH`"Q^XVJ*XV&`B!'
M&(VI*R!'&(VK*Z73C00LJ0"-``RR-%"P@1QB-4"P@1QBE^X4BI?R%(ZU0+$RB
M**X]+*P^+$P'&X;[A/Q,("M,("NI`8W-+(;[A/RI`(7]A?Z-4"R-Q"R@`+'[
MC:,L($<8C:0L($<8C:4L($<8C:8L($<8C:<L($<8C:@L($<8C5`L($<8($<8
MION&(J;\AB.M4"Q,\Q&-4"PX(/#_CLHLC,DLK%`LF$SD$*BI`(VJ+$Q4$1@@
M\/^E(J0C3!ZK&"#P_R"O%TJ-Q"RI%#CMQ"R%TR!LY4PH$Z``L2+P`\C0^8Q0
M+)A@(*\73$43(*\73&(3J*G_C:HL3%01("`8C<8L($<8C<<L($<8C<@L3*X3
M("``8C<LL($<8C<PL3`<4("``8C5PL($<8C5TL($<8C5XL($<8C5\L3%44AB*$
M(ZD!C<TL(*\73)X4AON$_*``L?M@AON$_(7](+W_I?T89?N%^ZD`9?R%_*``
ML?N-'2Q@(/0CH`"Q^V`@W1N,SBP@W1N,SRP@W1N,TBP@W1N,TRP@W1N,#BT@
MW1N,#"T@W1N,#RT@W1N,""T@W1N,$"T@W1N,"2T@W1N,"RT@W1N,"BU@K<XL
MC2TMK<\LC2XMK=(LC2\MK=,LC3`MK0XMC3DMK0PMC3\MK0\MC2(MK0@MC=TL
MK1`MC2@MC3DMK0LMC2'0K0HMC2#0HB2@+2`^&Z(IH"T@'!NB,:`M(/87HC6@
M+2`^&Z(ZH"T@+1NBU*`L(/87HMB@+"`M&ZT(+8V&`JD!A<>IWH4BJ2R%(Z[:
M+$R6%R"8&*D`C1$MC1(MC1,M(*@=CG\MC(`MC8$MJ0&%QZT_+8V&`JV!+4J-
M4"RI%#CM4"RHKCPM&"#P_Z``L2(@TO_(S($MD/6I`8T2+:D`C1$M(-T;C'@M
MC",M(*@=CE`LC'HMC7LMK1(M"AAM$BVJK5`LG7\MK7HMG8`MK7LMG8$MS1$M
MD`.-$2WN$BW.(RW0R2"H'8U&+<D`\`V@`+$BF4<MR,Q&+9#UK1$M&&D"2HU0
M+*D4..U0+(T=+8T8+8T4+:T1+1AI`6T=+8T>+8T9+8T5+:UX+1AI`DJ-4"RI
M#3CM4"R-'RV-&BV-%BVM>"T8:0%M'RV-("V-&RV-%RVM"2WP'LX8+>X9+<X:
M+>X;+:T0+8T<+<X4+>X5+<X6+>X7+<X4+<X5+>X6+>X7+:(4H"T@]A>M"2WP
M!Z(8H"T@/ANB':`M("T;K!TMR*X?+>@8(/#_K2(MC88"J0&%QZD`C2,MK2,M
M"AAM(RVJO8(MA2*]@RV%([V$+8U0+*``L2(@TO_(S%`LT/7FUJX=+>B&TR!L
MY>XC+:TC+<UX+9#$KA\MZ(Y`+:X=+8Y!+:X>+8Y"+:UX+8U#+:T/+8U$+:T,
M+8U%+:)`H"U,'Q<@J!V-4"R@`+$BF=XLR,Q0+)#UJ0"1(F"-#`..#0.,#@-@
MHN^@+XX-`XP.`V"&%(05H`>Q%)DK+(@0^$R7&X84A!6@!;$4F2LLB!#X3`H<
MAA2$%:`$L129*RR($/A,5!R&%(05H`6I`)DK+(@0^J``L12-+RS(L12-,"S(
ML12-*RS(L12-+2Q,71VI`(74C3<L(.8;(-T;KC<LF)TK+.XW+*TW+,D(D.RM
M+2R%UB!LY2`%'Z76*0&--RRL*RRN-RR]+RR1T;TQ+)'S[C<LK3<L*0&--RS(
MS"PLD.+PX"#^'*76S2XLD,SPRJD33-+_(/VN((JM3/>WH!BYV0`)@)G9`(@0
M]6"I`(U0+"#=&YBN4"R=*RSN4"S@!=#O(.8;I=9(I=-(KBTLH``8(/#_(`4?
MSBLLK"PLK2\LR?_P`I'1K3`LD?.(S"LLT.REULTN+/`&(/X<3"(<J1,@TO]H
MA=-HA=9,;.4@YANEUDBETTBN+2R@`!@@\/\@!1^L+"RM+RQ(*8`);I'1:)'S
MB*TO+$@I@`E`D=%HD?.(S"LLT.VM+RQ(*8`)<)'1:)'S(/X<I=;-+BRP':TO
M+$@I@`E=K"LLD=&L+"R1T6B1\ZPK+)'S3)H<K"PLK2\L2"F`"7V1T6B1\XBM
M+RQ(*8`)0)'1:)'SB,PK+-#MK2\L2"F`"6V1T6B1\VB%TVB%UB!LY4SF&^;6
MJ2@89=&%T:D`9=*%TJDH&&7SA?.I`&7TA?1@(-T;C"LL(-T;C"PL(-T;C"TL
M(-T;C"XL(-T;C"\LHBN@+$Q4'"#=&XPO+(TP+"#=&XPK+"#=&XPM+*D`C2PL
MC2XLH`(.*RPN+"R($/>@`@XM+"XN+(@0]ZTO+!AM+2R-H!VM,"QM+BR-H1VM
M*RP8:1>-G1VM+"QI+HV>'2`:(*``'N3DLF0#`B!#W3"(@(/VN()ZM(*.VIB*D
M(V"E>DBE>TB&>H1[((NPA4F$2FB%>VB%>F"%9(1E3%*JH$^I`)F_+8@0^F`@
MW1N,!RP@W1N,""P@W1N,"2P@W1N,"BQ@(.`=K`<LK@DLC@XL&"#P_R`%'QBE
MT6DHA?NETFD`A?P8I?-I*(7]I?1I`(7^K`<LL?N1T;']D?/(S`@L\/*0\.X.
M+*T.+,T*++`=&*71:2B%T:72:0"%TABE\VDHA?.E]&D`A?1,#!ZN"BRL!RP8
M3/#_(`4?(.`=K`<LK@HLC@XL&"#P_R`%'SBET>DHA?NETND`A?PXI?/I*(7]
MI?3I`(7^K`<LL?N1T;']D?/(S`@L\/*0\,X.+*T.+,T)+/`=.*71Z2B%T:72
MZ0"%TCBE\^DHA?.E].D`A?1,@1ZN"2RL!RP8(/#_3`4?(-T;C0PL(-T;C0TL
MK`<LK0PLD=&M#2R1\\C,""SP\)#N8*72..V(`AAIV(7TI=&%\V`@J!V.%RR,
M&"R-&2P@W1N,52R-5BP@W1N,&BP@`[D7+)'[B!#X3"(@J0&-#RRI
M`(W-+(U7+(U8+(U9+"#=&XQ5+(U6+*T>+/`7K54LC5<LK58LC5@LJ0"-52R-
M5BQ,LB<@^!\@&B"@`['[F1<LB!#X("(@K0\LT#>I28VH*ZU9+!AM5RR%_844
MJ0!M6"R%_H45KLTLT`,@*B"M&BR%_:D`A?ZI1HVH*Z[-+-`P("H@K1<LA?VM
M&"R%_J``(!H@L?V9ORW(S!DLD/7P\ZT9+(T4+"`B(*T/+/`#3.@G8*U5+(7]
MK58LA?X&_2;^!OTF_ABM/2QE_87[K3XL9?Z%_&!(J71XA0%H8$BI=X4!:%A@
MK<TL\`:I`(W-+&#JZNJE_4BE_DBI`(7]A?ZMJ"N%^ZDEA?RE>DBE>TBI^X5Z
MJ0"%>R"+L*``I?Z11\BE_9%':(5[:(5ZH`)HD5_(:)%?8*TK+(U5+*TL+(U6
M+&`@J!T@O?\@W1N,'2R8JJD!H``@NO\@W1NF%*05CALLAJZ,'"R$KZD`(,#_
M(,S_H@$@QO\@Y/\@Y/^@`"#D_R`:()&N("(@YJ[0`N:O(+?_*4+PZ*D!(,/_
M(,S_I:XX[1LLC3\LI:_M'"R-0"R@!$Y`+&X_+(@0]ZT_+#CI`HT_+(7]K4`L
MZ0"-0"R%_B`J(*T;+!AI((T;+*T<+&D`C1PL8"#=&XP]+(T^+&"I`(W-+(U!
M+(U"+(T>+(T/+"#=&XQ*+(P?+(P'+"#=&XQ++(P(+(P@+"#=&XQ,+(P)+(PA
M+"#=&XQ-+(P*+(PB+"#=&XQ&+"#=&XQ'+(PD+(P-+"#=&XQ(+"#=&XQ)+"#=
M&XQ7+(U8+"#=&XQ#+(U$+"#=&XQ%+*F@KD8LT`*I((TC+(T,+*(?H"P@+1NM
M32PX[4PLC4XLK4LL..U*+(U;+""3):D`(.3_\/O)+=`"J9W)*]`"J1W)(-`"
MJ0W)7]`"J87)5-`+KD4L\`8@/RL@^R/)$=`,($HD(+<B(/LC3-<AR9'0#"!*
M)"#S(B#[(TS7(<D3T`D@2B0@425,UR')D]`)($HD(&TE3-<AR1W0"2!*)"`G
M)4S7(<F=T`D@2B0@.R5,UR')A=`(K44L\`-,2B3)#=!-($HDK5DL&&U7+(7]
MJ0!M6"R% JU%+/`2(*,F(/LCJ1&-=P*I`87&3-<AJ0&-#RRM'BSP`TRR)R!?
```

```
M'ZU9+!AM5RR%_:D`;5@LA?Y,*B!,UR&M62S-3BRP!.Y9+&"M62P8;5<LA?NI
M`&U8+(7\I?S-1"R0"*7[S4,LD`%@(/P=(.\>[E<LT`/N6"Q,,".M62SP!,Y9
M+&"M6"P-5RS0`6"M0BS-6"R0":U!+,U7+)`!8"!Q'B#O'JU7+#CI`8U7+*U8
M+.D`C5@L3#`CK5DL&&U7+(U5+*D`;5@LC58LK1XL\`8@7R9,4",@>Q^M1BR%
MQZU'+*X:+/`#K4DLC88"K4PL&&U9+*JL2BS(&"#P_R`%'ZT7+(7[K1@LA?RL
M2BRMA@*1\\C,2RR0^/#VK1XLT!F@`"`:(+'[("(@(-+_R,Q;++`%S!DLD.I@
MK1<LA?NM&"R%_"`:(*``L?N%_2#T(['[*7^F_2`B("#-O2#T(ZD@(-+_(-+_
MH``@&B"Q^R`B("#2_\BETSCM'RS)'9#J8.;[T`+F_&"M'BSP`TR9)*U9+!AM
M5RR-52RI`&U8+(U6+"!['ZU,+!AM62RJK$PLR!@@\/\@!1^L2BRQT4F`D=&M
M2"RN&BSP`ZU)+)'SR,Q++)#G\.5@K1XL\`-,X"2M62P8;5<LC54LJ0!M6"R-
M5BP@>Q^M3"P8;5DLJJQ,+,@8(/#_(`4?K$HLL=%)@)'1K4<LKAHL\`.M22R1
M\\C,2RR0Y_#E8*U9+!AM5RR-52RI`&U8+(U6+"!?)JU,+!AM62RJK$PLR!@@
M\/\@!1^L2BRQT4F`D=&M2"RN&BSP`ZU)+)'SR,Q++)#G\.5@K5DL&&U7+(U5
M+*D`;5@LC58L(%\FK4PL&&U9+*JL3"S(&"#P_R`%'ZQ*++'128"1T:U'+*X:
M+/`#K4DLD?/(S$LLD.?PY6"M3BR-62R-42P@MR+.42S0^$S[(ZD`C5DLK4XL
MC5$L(/,BSE$LT/A,^R.M62SP"*D`C5DL3/LCK4$LC5<LK4(LC5@L3),EK5DL
MS4XLL`FM3BR-62Q,^R.M0RPX[4XLC5<LK40LZ0"-6"Q,DR6I`(U9+*(?H"P@
M+1L@,"/N62RM'BSP(ZU`+-`>K3\LS5DLL!:M62R-3BS.3BP8;0DLC0HLC4TL
M3-<EK5DLS4XLD,KPR*D`C5DL3/LCJ0&-'BRI`(W-+(T/+"#=&XQ*+(P?+(P'
M+)@8:1V-2RR-""R-("P@W1N,3"R,"2R,(2P@W1N,32R,"BR,(BP@W1N,1BP@
MW1N,1RR,)"R,#2P@W1N,2"P@W1N,22RI`*B,5RR,02R-6"R-0BRL/RRM0"R,
M0RR-1"P@W1N,12Q,J"&@!`Y5+"Y6+(@0]ZT;+!AM52R-%RRM'"QM5BR-&"RM
M%RR%^ZT8+(7\H`@&B"Q^XVF*\BQ^T@I?XVG*V@I@(T:+$PB(*T>+-`1(/@?
M(!H@H`.Q^TF`D?M,(B"M62P8;5<LC54LJ0!M6"R-5BP@7R8@&B"@`;'[28"-
M&BR1^TPB(""H'2"]_R#=&XP=+)BJJ0&@`""Z_R#=&Z84I!6I`"#5_Z``F)&N
M8"#=&R`:((P%+(T&+*``C'XMYA30`N85L130]N84T`+F%:44C3TLI16-/BRM
M!2R%%*T&+(45J0"-52R-5BRE%(T7+*45C1@L(!H@H`"Q%/``'R0WP`\C0]4B,
M&2S,?BV0`XQ^+:D`C1HL("(@(#`?[E4LT`/N5BQH\!NL&2S(F!AM%RR-%RR%
M%*D`;1@LC1@LA15,32<@(B"M52R%_:U6+(7^[GXMK7XMA11,*B"M62P8;5<L
MC54LJ0!M6"R-5BP@7R8@&B"B`(X4+*`"L?O(R2+0^;'[R2+P"IV_+>X4+.C(
MT/`@(B"MS2S05*(1H"P@MAVI%*`L(,X=(-4=J4*-J"NMIBN%_84BK:<KA?Z%
M(R`J(*E)C:@KK5DL&&U7+(7]A12I`&U8+(7^A14@*B"I1HVH*ZT:+(7]J0"%
M_DPJ(*U9+!AM5RR%%*D`;5@LA16MIBN%(JVG*X4CK1HLA?VI`(7^J0"-S2RB
MOZ`MK10L3`<;(-T;C!XL8"#=&XRJ*XR&`B#=&XRI*R#=&XRK*Z73C00LJ0"-
M`RR-%"P@J!W)`/`AC5`LS:LKD`.-JRN@`+$BF;\M(-+_[A0L[@,LR,Q0+)#L
MJ;Z%_:DMA?ZI`(T(W(W^*R#H*2#D_]`JK0C<S0(L\/.-`BSN_BNM_BO)`]#F
MJ0"-_BNE^_`&(``J3-HH(.@I3-HHC5`LR9W0`TP8*LD=T`-,-BK)E-`#3%(J
MKJLKT!2N%"S@)M`-K5`LC7<"J0&%QDP`*JX#+.P4+)`:KA0L\!7LJRN0$,D-
M\%')%/!8KJLK\`-,VBB@`+FL*\U0+/`(R,!2T/-,VBBM4"S)#?`LR13P,ZX#
M+.P4+)`#[A0L[@,LK5`LKJHKCH8"(-+_(.@IK`,LK5`LD?U,VBBM%"R%_"``
M*DSH)ZT#+-`#3-HHS10LL`8@NBI,VBBM%"S0`TS:*,X4+,X#+*F=(-+_J2`@
MTO\@TO^IG2#2_R#2_R#H*4S:**D!A?NM`RP8;00LJ+'1"8"1T:VI*Y'S8*D`
MA?NM`RP8;00LJ+'1*7^1T:VJ*Y'S8"``*JT#+,D!L`8@Z"E,VBC.`RP@Z"FI
MG2#2_TS:*"``*JT#+,T4+)`#3-HH[@,L(.@IJ1T@TO],VBBMJROP"ZT4+,VK
M*Y`#3-HHI?V%^:7^A?KF^=`"YOJL%"RQ_9'YB,P#+-#VK`,LR*D@D?VM!"R%
MTR!LY>X4+*T4+,DGD`6I)HT4+*`!L?T@TO_(S!0L\/60\ZT#+!AM!"R%TR!L
MY4S:**7]..D!A?FE_ND`A?JL`RS([A0LL?V1^<C,%"SP]I#TSA0LK00LA=,@
M;.7.`RR@`;']( -+_R,P4+-#UJ2`@TO_.%"RM`RP8;00LA=-,;.6M/2R%_:T^
M+(7^3"H@(-T;A/N%_""H'8U0+*``(!H@L2*1^R#T(\C,4"S0\ZD-D?L@(B!,
M]".M62R-HRNM5RR-I"NM6"R-I2NI`(U7+(U8+(U9+""C)NY7+-`#[E@LK5@L
MS40LD.VM5RS-0RR0Y?#CK:,KC5DLK:0KC5<LK:4KC5@L(),EK:,KC5DLK:0K
MC5<LK:4KC5@L3/LC`````!&!P$`04)#1$5&1TA)2DM,34Y/4%%24U155E=8
M65K!PL/$Q<;'R,G*R\S-SL_0T=+3U-76U]C9VB`Q,C,T-38W.#DP*RU<(2,D
M)28G*"E>*BXO.CT-%`````````````````````````!7)```ORT``````(
M`````````````````````````````!`>\O`````````````````````````
M`````````````````````````````````````````````````````````````
M````````````````````````````0$!``````!4````````````````````
M`````````0\)#@(&``4)"P(`"P4&#`````````````````````!?:0$"
M#PP")!<7`R46%J`#P]+3TB_2Q=35TLX@U$\@TT5,14-4`````````````````
M`````````!P$```$/`P,$`````````````````*`#```G`!@#`28!
M%U]I`PX"(P,%`R0"!`$$(P,#H`$`````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M````````````$*))!I8))%"B'-P<(AG!P>!&!AF9A@8@8!D1@@08B8!!Q@@
M,`P$&.`D),,8&,,D)%6J5:I5JE6JS#/,,\PSS#-F@68\9H%F/%K;&.?G&-M:
M$1$B(D1$B(@1$2(R3$2(B,`P#`/`,`P#P#`,"]`P#`,1$:JJ1$2JJO5$1$!?
M1$0$B%`@4(@```J5:I5JE6J5:JJJJI55555`@H(*""@@((!`0$"`@49X0$"
M'.`!`AS@B!$B1(A$(A$B1(A$(D2(1`000@@A!!``0*1*!"!2)0)!(A2(%")!
M"$$JE"*4*D$4@1@89F88&($!)F(0"$9D@(/&;#@8#`8#*!""1((0*`#=$7=$
MW1%W1`Q@`QC`!C"!4HA2`"6()0`4$LDDDT@H*!ADK()J3#``_\#>WL;V]@9T
M(A>[<2)'[G6NU;I7ZEVKP]L8?GX8V\.`1`@0($0"`?__________`1$0?!`1
M`<<1$1'^$1$1[_]@9F9F9F8&P_`\#\/P/`\0.$2"`8)$.(1F2!`DS$(!<(@'
M`'"(!P`B290B22*42:2DI*1*2DI*$43N1!%$[D0!;6T0;6T![J2DI.1*2DI.
@*`!4`"@``0`2"81"0H0)$D2(1(A$B$2(B@4B4*A0(@50
`
end

@(A)menucode2: Menu Toolbox at $8000 (32768)
```

```
begin 644 mbox8000.bin
M`(!,>8M,](M,&XU,0(U,$8M,WY9,^8U,;HY,XXY,)Y%,%8],0X],!I=,AI!,
M,9%,WY5,=IA,?9A,"9M,%IM,'9M,TX%,SX!,3(%,O81,$H-,<8-,.X-,6(-,
ME(-,G(-,^X-,/81,FX1,'(M,+8M,/HM,3XM,RX1,]H1,!X5,&(5,(85,*(5,
M7(5,>(5,B(5,K85,.H9,P(9,Q(9,!X=,$(=,%X=,&H=,;H=,@H=,BX=,EH=,
MO8=,PX=,R8=,TH=,YX=,]H=,$8A,)HE,3XA,[XHX(/#_CLJ<C,F<(-V+A/R8
MP`2P`6`8:02%_JD`A?V%^ZD3(-+_(`6/>*D`A0&B`Z``L=&1^\8!L?/F`9']
MR-#QYM+F_.;TYO[*$.;&_J#IQ@&M(-#F`9']R,8!K2'0Y@&1_<BMRIR1_<BM
MR9R1_:EWA0%8KLJ<K,F<&$SP_ZD`C:J<(-V+A/R8&&D$A?ZI`(7]A?NI$R#2
M_R`%CWBI`(4!H@.@`+'1C5"<+*J<$`FM4)PIO\D@T`RQ^Y'1L?W&`9'SY@'(
MT-[FTN;\YO3F_LH0T\;^+*J<,!^@Z;']Q@&-(-#(Y@&Q_<8!C2'0Y@'(L?V%
MULBQ_873J7>%``:D`C:J<6$QLY:D`A?V%_HU0G(W$G"#=BYBL4)R9HYSN4)S`
M!=#O(*B-R1F0`JD9C<6<H`"Q(IEFG,C,Q9S0]2"]A*ZCG*``C+N<&"#P_R`%
MCZVHG(V\G"#<@B#D__#[H@"-Q)RNNYR.4)S)$?`0R9'P'R#U@LD-\"_0`$PD
M@NZ[G*V[G,VFG-!2J0"-NYQ,I8+.NYRMNYS)__`#3*6"K*:<B(R[G$RE@JW$
MG/`&K5"<3(""K;N<&&VCG*J@`"#P_R`%CZVGG(V\G"#<@JV[G!AI`87]J0"%
M_DPJD*VCG!AMNYRJH``8(/#_(`6/K:B<C;R<(-R"K:.<&&U0G*J@`!@@\/\@
M!8^MIYR-O)P@W(),)(*LI)RMO)R1\S`&L=%)@)'1R,REG/#MD.M@JIA(H`"*
MV6:<T`B,NYRB#8[$G,C,Q9S0[6BHBF`@O80@W8N$TR#=BX36(&SE(*B-C5"<
MK5"<\`V@`+$B(-+_R,Q0G-#U8""HC<D`\!6-4)R@`+$BR4&0!`F`D2+(S%"<
MT/!@(*B-C5"<R0#P#J``L2(I?Y$BR,Q0G-#T8"#=BYBJH``8(/#_(*B-C5"<
M2HW$G*D4..W$G(73(&SE3"B#J?^-JIQ,48$@W8N,QIP@W8N,QYP@W8N,R)P@
MO82I$R#2_R`%CZD8A?N@)['1S<:<T`RMQYR1T:W(G#`"D?.($.K&^S`=J2@8
M9=&%T:D`9=*%TABI*&7SA?.I`&7TA?1,O8.I$TS2_R#=BXS+G"#=BXS,G""]
MA*D3(-+_(`6/J1B%^Z`GL?,I#\W+G-`%K<R<D?.($._&^S#)&*DH9?.%\ZD`
M9?2%]$P6A"#=BXQ<G"#=BXQ=G"#=BXQ>G"#=BXQ?G""]A*762*732*Y>G*``
M&"#P_R`%C\Y<G*Q=G+'S*0^JO:N<D?.(S%R<T/"EULU?G/`+YM8@;.4@!8],
M;81HA=-HA=9,;.4@J(V%^R#D__#[H`#1(M`+R)B%_:D`A?Y,*I#(Q/O0[/#C
MH!BYV0`)@)G9`(@0]6!(J0&-S9QH("F(JJD!H``@NO\@1XB%%"!'B(45IA2D
M%:D`(-7_H`"8D:Y@AON$_*`#L?N9!YR($/A,_(V&^X3\H`.Q^YD'G(@0^$QQ
MCHX,G(P-G$SOCHX]G(P^G&`@&I".%YR,&)R-&9R&^X3\&&7[A?NI`&7\A?R@
M`+'[C56<($>(C5:<($>(C1J<("*03#"/CE6<C%:<J0&-S9RI`(T/G(U7G(U8
MG(U9G$Q?CX;[A/RD^Z7\H@&.S9Q,"9=(J0&-S9QH("F(KAV<J0&@`""Z_R!'
MB(44($>(A16F%*053**0J0&-S9R&^X3\H``@&I"I`(T>G(T/G+'[C4J<C1^<
MC0><($>(C4N<C0B<C2"<($>(C4R<C0F<C2&<($>(C4V<C0J<C2*<($>(C4:<
M($>(C4><C22<C0V<($>(C4B<($>(C4F<($>(C5><C4&<($>(C5B<C4*<($>(
MC4.<($>(C42<($>(C46<("*03*B1AON$_"`:D*D!C1Z<J0"-#YR@`+'[C4J<
MC1^<C0><&&D=C4N<C0B<C2"<($>(C4R<C0F<C2&<($>(C4V<C0J<C2*<($>(
MC4:<($>(C4><C22<C0V<($>(C4B<($>(C4F<J0"-5YR-09R-6)R-0IRN/YRM
M0)R.0YR-1)P@1XB-19P@(I!,J)&-'IQ@AON$_*D!C<V<H`"Q^XVJFXV&`B!'
MB(VIFR!'B(VKFZ73C02<J0"-`YR-%)P@1XB-4)P@1XBE^X4BI?R%(ZU0G$RB
MF*X]G*P^G$P'BX;[A/Q,()M,()NI`8W-G(;[A/RI`(7]A?Z-4)R-Q)R@`+'[
MC:.<($>(C:2<($>(C:6<($>(C::<($>(C:><($>(C:B<($>(C5"<($>((($>(
MION&(J;\AB.M4)Q,\X&-4)PX(/#_CLJ<C,F<K%"<F$SD@*BI`(VJG$Q4@1@@
M\/^E(J0C3!ZK&"#P_R"OATJ-Q)RI%#CMQ)R%TR!LY4PH@Z``L2+P`\C0^8Q0
MG)A@(*^'3$6#(*^'3&*#J*G_C:J<3%2!(""(C<:<($>(C<><($>(C<B<3*Z#
M(""(C<N<($>(C<R<3`>$(""(C5R<($>(C5V<($>(C5Z<($>(C5^<3%6$AB*$
M(ZD!C<V<(*^'3)Z$AON$_*``L?M@AON$_(7](+W_I?T89?N%^ZD`9?R%_*``
ML?N-'9Q@(/23H`"Q^V`@W8N,SIP@W8N,SYP@W8N,TIP@W8N,TYP@W8N,#IT@
MW8N,#)T@W8N,#YT@W8N,")T@W8N,$)T@W8N,"9T@W8N,"YT@W8N,"IU@K<Z<
MC2V=K<^<C2Z=K=*<C2^=K=.<C3"=K0Z=C3F=K0R=C3^=K0^=C2*=K0B=C=V<
MK1"=C2B=C3F=K0N=C2'0K0J=C2#0HB2@G2`^BZ(IH)T@'(NB,:"=(/:'HC6@
MG2`^BZ(ZH)T@+8NBU*"<(/:'HMB@G"`MBZT(G8V&`JD!A<>IWH4BJ9R%(Z[:
MG$R6AR"8B*D`C1&=C1*=C1.=(*B-CG^=C("=C8&=J0&%QZT_G8V&`JV!G4J-
M4)RI%#CM4)RHKCR=&"#P_Z``L2(@TO_(S(&=D/6I`8T2G:D`C1&=(-V+C'B=
MC".=(*B-CE"<C'J=C7N=K1*="AAM$IVJK5"<G7^=K7J=G8"=K7N=G8&=S1&=
MD`.-$9WN$IW.(YW0R2"HC8U&G<D`\`V@`+$BF4>=R,Q&G9#UK1&=&&D"2HU0
MG*D4..U0G(T=G8T8G8T4G:T1G1AI`6T=G8T>G8T9G8T5G:UXG1AI`DJ-4)RI
M#3CM4)R-'YV-&IV-%IVM>)T8:0%M'YV-()V-&YV-%YVM"9WP'LX8G>X9G<X:
MG>X;G:T0G8T<G<X4G>X5G<X6G>X7G<X4G<X5G>X6G>X7G:(4H)T@]H>M"9WP
M!Z(8H)T@/HNB':"=("V+K!V=R*X?G>@8(/#_K2*=C88"J0&%QZD`C2.=K2.=
M"AAM(YVJO8*=A2*]@YV%([V$G8U0G*``L2(@TO_(S%"<T/7FUJX=G>B&TR!L
MY>XCG:TCG<UXG9#$KA^=Z(Y`G:X=G8Y!G:X>G8Y"G:UXG8U#G:T/G8U$G:T,
MG8U%G:)`H)U,'X<@J(V-4)R@`+$BF=Z<R,Q0G)#UJ0"1(F"-#`..#0.,#@-@
MHN^@GXX-`XP.`V"&%(05H`>Q%)DKG(@0^$R7BX84A!6@!;$4F2N<B!#X3`J,
MAA2$%:`$L129*YR($/A,5(R&%(05H`6I`)DKG(@0^J``L12-+YS(L12-,)S(
ML12-*YS(L12-+9Q,78VI`(74C3><(.:+(-V+KC><F)TKG.XWG*TWG,D(D.RM
M+9R%UB!LY2`%CZ76*0&--YRL*YRN-YR]+YR1T;TQG)'S[C><K3><*0&--YS(
MS"R<D.+PX"#^C*76S2Z<D,SPRJD33-+_(/VN((JM3/>WH!BYV0`)@)G9`(@0
M]6"I`(U0G"#=BYBN4)R=*YSN4)S@!=#O(.:+I=9(I=-(KBV<H``8(/#_(`6/
MSBN<K"R<K2^<R?_P`I'1K3"<D?.(S"N<T.REULTNG/`&(/Z,3"*,J1,@TO]H
MA=-HA=9,;.4@YHNEUDBETTBN+9R@`!@@\/\@!8^L+)RM+YQ(*8`);I'1:)'S
MB*TOG$@I@`E`D=%HD?.(S"N<T.VM+YQ(*8`)<)'1:)'S(/Z,I=;-+IRP':TO
MG$@I@`E=K"N<D=&L+)R1T6B1\ZPKG)'S3)J,K"R<K2^<2"F`"7V1T6B1\XBM
M+YQ(*8`)0)'1:)'SB,PKG-#MK2^<2"F`"6V1T6B1\VB%TVB%UB!LY4SFB^;6
MJ2@89=&%T:D`9=*%TJDH&&7SA?.I`&7TA?1@(-V+C"N<(-V+C"R<(-V+C"V<
M(-V+C"Z<(-V+C"^<HBN@G$Q4C"#=BXPOG(TPG"#=BXPKG"#=BXPMG*D`C2R<
MC2Z<H`(.*YPN+)R($/>@`@XMG"XNG(@0]ZTOG!AM+9R-H(VM,)QM+IR-H8VM
M*YP8:1>-G8VM+)QIGHV>C2`:D*``'N3F<F0#`B!#W3"*0(/VN()ZM(*.VIB*D
M(V"E>DBE>TB&>H1[((NPA4F$2FB%>VB%>F"%9(1E3%*JH$^I`)F_G8@0^F`@
MW8N,!YP@W8N,")P@W8N,"9P@W8N,"IQ@(."-K`><K@F<C@Z<&"#P R`%CQBE
```

```
MT6DHA?NETFD`A?P8I?-I*(7]I?1I`(7^K`><L?N1T;']D?/(S`B<\/*0\.X.
MG*T.G,T*G+`=&*71:2B%T:72:0"%TABE\VDHA?.E]&D`A?1,#(ZN"IRL!YP8
M3/#_(`6/(."-K`><K@J<C@Z<&"#P_R`%CSBET>DHA?NETND`A?PXI?/I*(7]
MI?3I`(7^K`><L?N1T;']D?/(S`B<\/*0\,X.G*T.G,T)G/`=.*71Z2B%T:72
MZ0"%TCBE\^DHA?.E].D`A?1,@8ZN"9RL!YP8(/#_3`6/(-V+C0R<(-V+C0V<
MK`><K0R<D=&M#9R1\\C,")SP\)#N8*72..V(`AAIV(7TI=&%\V`@J(V.%YR,
M&)R-&9P@W8N,59R-5IP@W8N,&IP@^(\@&I"@`[D7G)'[B!#X3"*0J0&-#YRI
M`(W-G(U7G(U8G(U9G"#=BXQ5G(U6G*T>G/`7K56<C5><K5:<C5B<J0"-59R-
M5IQ,LI<@^(\@&I"@`['[F1><B!#X("*0K0^<T#>I28VHFZU9G!AM5YR%_844
MJ0!M6)R%_H45KLV<T`,@*I"M&IR%_:D`A?ZI1HVHFZ[-G-`P("J0K1><A?VM
M&)R%_J``(!J0L?V9OYW(S!F<D/7P\ZT9G(T4G"`BD*T/G/`#3.B78*U5G(7]
MK5:<A?X&_2;^!OTF_ABM/9QE_87[K3Z<9?Z%_&!(J71XA0%H8$BI=X4!:%A@
MK<V<\`:I`(W-G&#JZNJE_4BE_DBI`(7]A?ZMJ)N%^ZDEA?RE>DBE>TBI^X5Z
MJ0"%>R"+L*``I?Z11\BE_9%':(5[:(5ZH`)HD5_(:)%?8*TKG(U5G*TLG(U6
MG&`@J(T@O?\@W8N,'9R8JJD!H``@NO\@W8NF%*05CAN<AJZ,')R$KZD`(,#_
M(,S_H@$@QO\@Y/\@Y/^@`"#D_R`:D)&N("*0YJ[0`N:O(+?_*4+PZ*D!(,/_
M(,S_I:XX[1N<C3^<I:_M')R-0)R@!$Y`G&X_G(@0]ZT_G#CI`HT_G(7]K4"<
MZ0"-0)R%_B`JD*T;G!AI((T;G*T<G&D`C1R<8"#=BXP]G(T^G&"I`(W-G(U!
MG(U"G(T>G(T/G"#=BXQ*G(P?G(P'G"#=BXQ+G(P(G(P@G"#=BXQ,G(P)G(PA
MG"#=BXQ-G(P*G(PBG"#=BXQ&G"#=BXQ'G(PDG(P-G"#=BXQ(G"#=BXQ)G"#=
MBXQ7G(U8G"#=BXQ#G(U$G"#=BXQ%G*F@KD:<T`*I((TCG(T,G*(?H)P@+8NM
M39PX[4R<C4Z<K4N<..U*G(U;G""3E:D`(.3_\/O)+=`"J9W)*]`"J1W)(-`"
MJ0W)7]`"J87)5-`+KD6<\`8@/YL@^Y/)$=`,($J4(+>2(/N33->1R9'0#"!*
ME"#SDB#[DTS7D<D3T`D@2I0@495,UY')D]`)($J4(&V53->1R1W0"2!*E"`G
ME4S7D<F=T`D@2I0@.Y5,UY')A=`(K46<\`-,2I3)#=!-($J4K5F<&&U7G(7]
MJ0!M6)R%_JU%G/`2(*.6(/N3J1&-=P*I`87&3->1J0&-#YRM'ISP`TRRER!?
MCZU9G!AM5YR%_:D`;5B<A?Y,*I!,UY&M69S-3IRP!.Y9G&"M69P8;5><A?NI
M`&U8G(7\I?S-1)R0"*7[S4.<D`%@(/R-(.^.[E><T`/N6)Q,,).M69SP!,Y9
MG&"M6)P-5YS0`6"M0IS-6)R0":U!G,U7G)`!8"!QCB#OCJU7G#CI`8U7G*U8
MG.D`C5B<3#"3K5F<&&U7G(U5G*D`;5B<C5:<K1Z<\`8@7Y9,4),@>X^M1IR%
MQZU'G*X:G/`#K4F<C88"K4R<&&U9G*JL2IS(&"#P_R`%CZT7G(7[K1B<A?RL
M2IRMA@*1\\C,2YR0^/#VK1Z<T!F@`"`:D+'[("*0(-+_R,Q;G+`%S!F<D.I@
MK1><A?NM&)R%_"`:D*``L?N%_2#TD['[*7^F_2`BD"#-O2#TDZD@(-+_(-+_
MH``@&I"Q^R`BD"#2_\BETSCM'YS)'9#J8.;[T`+F_&"M'ISP`TR9E*U9G!AM
M5YR-59RI`&U8G(U6G"![CZU,G!AM69RJK$R<R!@@\/\@!8^L2IRQT4F`D=&M
M2)RN&ISP`ZU)G)'SR,Q+G)#G\.5@K1Z<\`-,X)2M69P8;5><C56<J0!M6)R-
M5IP@>X^M3)P8;5F<JJQ,G,@8(/#_(`6/K$J<L=%)@)'1K4><KAJ<\`.M29R1
M\\C,2YR0Y_#E8*U9G!AM5YR-59RI`&U8G(U6G"!?EJU,G!AM69RJK$R<R!@@
M\/\@!8^L2IRQT4F`D=&M2)RN&ISP`ZU)G)'SR,Q+G)#G\.5@K5F<&&U7G(U5
MG*D`;5B<C5:<(%^6K4R<&&U9G*JL3)S(&"#P_R`%CZQ*G+'128"1T:U'G*X:
MG/`#K4F<D?/(S$N<D.?PY6"M3IR-69R-49P@MY+.49S0^$S[DZD`C5F<K4Z<
MC5&<(/.2SE&<T/A,^Y.M69SP"*D`C5F<3/N3K4&<C5><K4*<C5B<3).5K5F<
MS4Z<L`FM3IR-69Q,^Y.M0YPX[4Z<C5><K42<Z0"-6)Q,DY6I`(U9G*(?H)P@
M+8L@,)/N69RM'ISP(ZU`G-`>K3^<S5F<L!:M69R-3IS.3IP8;0F<C0J<C4V<
M3->5K5F<S4Z<D,KPR*D`C5F<3/N3J0&-'IRI`(W-G(T/G"#=BXQ*G(P?G(P'
MG)@8:1V-2YR-")R-()P@W8N,3)R,"9R,(9P@W8N,39R,"IR,(IP@W8N,1IP@
MW8N,1YR,))R,#9P@W8N,2)P@W8N,29RI`*B,5YR,09R-6)R-0IRL/YRM0)R,
M0YR-1)P@W8N,19Q,J)&@!`Y5G"Y6G(@0]ZT;G!AM59R-%YRM')QM5IR-&)RM
M%YR%^ZT8G(7\H``@&I"Q^XVFF\BQ^T@I?XVGFV@I@(T:G$PBD*T>G-`1(/B/
M(!J0H`.Q^TF`D?M,(I"M69P8;5><C56<J0!M6)R-5IP@7Y8@&I"@`;'[28"-
M&IR1^TPBD""HC2"]_R#=BXP=G)BJJ0&@`""Z_R#=BZ84I!6I`"#5_Z``F)&N
M8"#=BR`:D(P%G(T&G*``C'Z=YA30`N85L130]N84T`+F%:44C3V<I16-/IRM
M!9R%%*T&G(45J0"-59R-5IRE%(T7G*45C1B<(!J0H`"Q%/``R0WP`\C0]4B,
M&9S,?IV0`XQ^G:D`C1J<("*0(#"/[E6<T`/N5IQH\!NL&9S(F!AM%YR-%YR%
M%*D`;1B<C1B<A15,39<@(I"M59R%_:U6G(7^[GZ=K7Z=A11,*I"M69P8;5><
MC56<J0!M6)R-5IP@7Y8@&I"B`(X4G*``L?O(R2+0^;'[R2+P"IV_G>X4G.C(
MT/`@(I"MS9S05*(1H)P@MHVI%*"<(,Z-(-6-J4*-J)NMIIN%_84BK:>;A?Z%
M(R`JD*E)C:B;K5F<&&U7G(7]A12I`&U8G(7^A14@*I"I1HVHFZT:G(7]J0"%
M_DPJD*U9G!AM5YR%%*D`;5B<A16MIIN%(JVGFX4CK1J<A?VI`(7^J0"-S9RB
MOZ"=K12<3`>+(-V+C!Z<8"#=BXRJFXR&`B#=BXRIFR#=BXRKFZ73C02<J0"-
M`YR-%)P@J(W)`/`AC5"<S:N;D`.-JYN@`+$BF;^=(-+_[A2<[@.<R,Q0G)#L
MJ;Z%_:F=A?ZI`(T(W(W^FR#HF2#D_]`JK0C<S0*<\/.-`ISN_INM_IO)`]#F
MJ0"-_INE^_`&(`":3-J8(.B93-J8C5"<R9W0`TP8FLD=T`-,-IK)E-`#3%*:
MKJN;T!2N%)S@)M`-K5"<C7<"J0&%QDP`FJX#G.P4G)`:KA2<\!7LJYN0$,D-
M\%')%/!8KJN;\`-,VIB@`+FLF\U0G/`(R,!2T/-,VIBM4)S)#?`LR13P,ZX#
MG.P4G)`#[A2<[@.<K5"<KJJ;CH8"(-+_(.B9K`.<K5"<D?U,VIBM%)R%_"``
MFDSHEZT#G-`#3-J8S12<L`8@NII,VIBM%)S0`TS:F,X4G,X#G*F=(-+_J2`@
MTO\@TO^IG2#2_R#2_R#HF4S:F*D!A?NM`YP8;02<J+'1"8"1T:VIFY'S8*D`
MA?NM`YP8;02<J+'1*7^1T:VJFY'S8"``FJT#G,D!L`8@Z)E,VIC.`YP@Z)FI
MG2#2_TS:F"``FJT#G,T4G)`#3-J8[@.<(.B9J1T@TO],VIBMJYOP"ZT4G,VK
MFY`#3-J8I?V%^:7^A?KF^=`"YOJL%)RQ_9'YB,P#G-#VK`.<R*D@D?VM!)R%
MTR!LY>X4G*T4G,DGD`6I)HT4G*`!L?T@TO_(S!2<\/60\ZT#G!AM!)R%TR!L
MMY4S:F*7]..D!A?FE_ND`A?JL`YS([A2<L?V1^<C,%)SP]I#TSA2<K02<A=,@
M;.7.`YR@`;']('-+_R,P4G-#UJ2`@TO_.%)RM`YP8;02<A=-,;.6M/9R%_:T^
MG(7^3"J0(-V+A/N%_""HC8U0G*``(!J0L2*1^R#TD\C,4)S0\ZD-D?L@(I!,
M]).M69R-HYNM5YR-I)NM6)R-I9NI`(U7G(U8G(U9G""CEN7G-`#[EB<K5B<
MS42<D.VM5YS-0YR0Y?#CK:.;C5F<K:2;C5><K:6;C5B<().5K:.;C5F<K:2;
MC5><K:6;C5B<3/N3`````!&!P$`04)#1$5&1TA)2DM,34Y/4%%24U155E=8
M65K!PL/$Q<;'R,G*R\S-SL_0T=+3U-76U]C9VB`Q,C,T-38W.#DP*RU<(2,D
M)28G*"E>*BXO.CT-%`````````````````````!7)```OYT``````(
M`````````````````````````````!`>^?`````````````
```

```
M`````````````````````````````````````````````````````````````
M`````````````````````````````0$!``````!4```````````````````
M`````````0\)#@(&``4)"P(`"P4&#`````````````````````````!?:0$"
M#PP")!<7`R46%J`#P]+3TB_2Q=35TLX@U$\@TT5,14-4```````````````````
M`````````````!P$```$/`P,$`````````````````````*`#```G`!@#`28!
M%U]I`PX"(P,%`R0"!`$$(P,#H`$`````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M`````````````````````````````````````````````````````````````
M````````````$*))!I8))%"B'-P<(AG!P>!&!AF9A@8@8!D1@@08B8!!Q@@
M,`P$&.`D),,8&,,D)%6J5:I5JE6JS#/,,\PSS#-F@68\9H%F/%K;&.?G&-M:
M$1$B(D1$B(@1$2(R3$2(B,`P#`/`,`P#P#`,"]`P#`,1$:JJ1$2JJO5$1$!?
M1$0$B%`@4(@```"J5:I5JE6J5:JJJJI55555`@H(*""@@((!`0$"`@49X0$"
M'.`!`AS@B!$B1(A$(A$B1(A$(D2(1`000@@A!!``0*1*!"!2)0)!(A2(%")!
M"$$JE"*4*D$4@1@89F88&($!)F(0"$9D@(/&;#@8#`8#*!""1((0*`#=$7=$
MW1%W1`Q@`QC`!C"!4HA2`"6()0`4$LDDDT@H*!ADK()J3#``_\#>WL;V]@9T
M(A>[<2)'[G6NU;I7ZEVKP]L8?GX8V\.`1`@0($0"`?__________`1$0?!`1
M`<<1$1'^$1$1[_]@9F9F9F8&P_`\#\/P/`\0.$2"`8)$.(1F2!`DS$(!<(@'
M`'"(!P`B290B22*42:2DI*1*2DI*$43N1!%$[D0!;6T0;6T![J2DI.1*2DI.
@*`!4`"@``0`2"81"0H0)$D2(1(A$B$2(B@4B4*A0(@50
`
end

========================================================================
@(#): bottom
```